Programming manual

S-SDK-BTS256

Document created: 2021-1-26

Kapitel 1

Information

Please read this documentation and the disclaimer carefully before using the software.
By installing and using the software, you explicitly and fully acknowledge and agree to this.
Gigahertz-Optik GmbH reserves the right to make changes to this manual without prior notice.

1.1 Disclaimer

This software was developed with utmost care and thoroughly tested on different computers. No errors were noted for the approved product versions. However, it cannot be guaranteed that the software will work perfectly on all types of computers. Completely error-free software is not possible with the current technology level.
Gigahertz-Optik GmbH is not liable if the software does not perfectly fulfill your desired purpose or if it is incompatible with other software on your computer. You are therefore solely responsible for the choice, installation and use as well as for the intended results.
With the exception of damages caused deliberately, Gigahertz-Optik GmbH is not liable for any damages caused by the use or inability to use the software. This also exclusively applies for loss of business profits, business interruptions, loss of business information or any other economic losses, even if Gigahertz-Optik GmbH had been previously advised of the possibility of such damage. The enclosed documentation/help of the software is with no claim of accuracy or completeness.

1.2 Warranty

Gigahertz-Optik GmbH guarantees the delivery of all functions listed in the product description. Any available delivery media are free of any material defects.
We have taken all the necessary and possible steps that are required to keep this software free of viruses, spyware, the so-called "back door entrancesâ€ť or other harmful code. We do not collect any information about you or your data. We will not deliberately limit you from using the functions of this software or access to your data. This agreement supersedes any non-contractual assurances that we may have explained to you. Any modification to this agreement must be confirmed in writing by both parties.

1.3 License

A license for the full version allows you to use the product on only one workstation. Each concurrent use on another workstation requires an additional product license. The distribution of the product and documentation is prohibited. You are authorized to make a copy of this product for your backup purposes. You may pass on your own software that you have developed using this development package together with the required DLLs for this development package to third parties.

1.4 Overview

This development package provides you with all the tools required (no compiler or integrated software development environments) to directly control a BTS256 series measurement device from Gigahertz-Optik using C/C++. This is primarily with regards to the communication and control libraries for your BTS256.
In order to use these libraries, you need a programming environment such as Microsoft Visual Studio, Embarcadero C++ builder, etc.

1.5 Contact information of Gigahertz-Optik

Gigahertz Optik GmbH	Gigahertz-Optik Inc
An der Kälberweide 12 D-82299 Türkenfeld Germany Tel.: +49 8193 93700-0 Fax: +49 8193 93700-50 Email: info@gigahertz-optik.de Homepage: http://www.gigahertz-optik.de	110 Haverhill Road Amesbury MA 01913 USA Tel: + 978 462 1818 Fax: + 978 462 3677 Email: b.angelo@gigahertz-optik.com Homepage: https://www.gigahertz-optik.com

1.6 System Requirements

To use the S-SDK BTS256 you have to consider the following points:

- Minimum disk space – approx. 10MB
- Operation system: MS Windows XP, MS Windows 7 (32bit/64bit), MS Windows 10 (32bit/64bit)
- C/C++ development environment such as MS Visual Studio, Embarcadero C++ Builder, etc. when programming with C/C++
- free USB port

1.7 Installation

Follow the steps below to install the BTS256-SDK from the product CD:

- Read this documentation before you begin the installation

- Close all other applications before installing
- Insert the CD in your CD drive or unpack the supplied ZIP file.
- Copy the Gigahertz-Optik folder from the CD or ZIP file to a location of your choice. If you have already got other development packages from Gigahertz-Optik installed, it is recommended to use the same installation path in order to avoid possible conflicts.
- Add the folder "install dir/Gigahertz-Optik/runtime"t' to your system path. "install dir" hereby corresponds to the base path you added in step 4 above. If you have already got other development packages from Gigahertz-Optik installed, step 5 might not be necessary.

1.8 System preparation

Connect the BTS256 to your computer. The required drivers are standard windows drivers and will be installed automatically.
The SDK is password protected! Each programm created with this SDK has to call the function setPassword() before any other function can be called.

Kapitel 2

Programming example

Example - How to import DLL in your application

Note

The method descriptions (Module) provide examples of how to use the SDK methods.

2.1 C++

As we don't deliver import libraries for different development environments you have to use run-time dynamic linking to be able to use all methods provided by dll.
Following is an C++ example of how to import and use methods from DLL. The example does not include all available methods. The use of the handles is encapsulated in the class. The example searches and initializes a BTS256, peforms a measurement and then records the results in the console.
At the end, all BTS256-resources are released again.

2.1.1 BTS256Example.cpp

```
#include "BTS256Import.h
#include <iostream>
int main(int argc, char* argv[])
{
    BTS256Import bts256;
    //search for a BTS256 device
    //first you have to replace the right password in the BTS256Import.cpp
    int error = bts256.init("BTS256_0
");
    if (error == 0)
    {
        char userinput[10];
        //write all available calibration entries to the console
        bts256.writeCalibrationInfoToConsole();
        //let the user choose a calibration
        std::cout « "Please choose a calibration number:
;
        std::cin.getline(userinput, 10);
        bts256.setCalibrationEntry(atoi(userinput));
        //set measurement mode and start a new measurement
        //spectralIntegrationTime = 50ms
        bts256.setSpectralMeasurementSettings(50000);
        error = bts256.measure();

        //if no error occured read the integral values
        if (error == 0)
        {
            double value;
            char unit[256];
            bts256.integralGetValues(&value, unit);
```

```
            std::cout « ïntegral sensor =
 « value « "
 « unit « std::endl;
        }
        else
        {
            std::cout « ërror occured:
 « error « std::endl;
        }
        bts256.close();
    }
    else
    {
        std::cout « ërror occured:
 « error « std::endl;
    }
    system("PAUSE
);
}
```

2.1.2 BTS256Import.cpp

```
#include "BTS256Import.h
BTS256Import::BTS256Import()
{
    hDLLGOBTS256 = NULL;
    handle = -1;
}
BTS256Import:: BTS256Import()
{
}
int __stdcall BTS256Import::init(char* deviceName)
{
    int l_rc = 0;
    if (handle > 0)
        close();
    if (getProcAddresses(&hDLLGOBTS256, "GOMDBTS256.dll
, 10,
        &GOMDBTS256_setPassword, "GOMDBTS256_setPassword
,
        &GOMDBTS256_getHandle, "GOMDBTS256_getHandle
,
        &GOMDBTS256_releaseHandle, "GOMDBTS256_releaseHandle
,
        &GOMDBTS256_setCalibrationEntryNumber, "GOMDBTS256_setCalibrationEntryNumber
,
        &GOMDBTS256_getSelectedCalibrationEntryNumber, "GOMDBTS256_getSelectedCalibrationEntryNumber
,
        &GOMDBTS256_readCalibrationEntryInfo, "GOMDBTS256_readCalibrationEntryInfo
,
        &GOMDBTS256_measure, "GOMDBTS256_measure
,
        &GOMDBTS256_getCWValue, "GOMDBTS256_getCWValue
,
        &GOMDBTS256_integralGetUnit, "GOMDBTS256_integralGetUnit
,
        &GOMDBTS256_spectralSetIntegrationTimeInMs, "GOMDBTS256_spectralSetIntegrationTimeInMs
        ))
    {
        try {
            l_rc = GOMDBTS256_setPassword("passw
); //replace passw with the right password
            if (l_rc == 0)
                l_rc = GOMDBTS256_getHandle(deviceName, &handle);
        }
        catch (...) {
            l_rc = -1;
        }
    }
    else {
        l_rc = -1;
    }
    return l_rc;
}
int __stdcall BTS256Import::writeCalibrationInfoToConsole()
{
    char calibInfo[100];
    std::cout « Ävailable calibration entries:
 « std::endl;
    for (int i = 0; i < 52; i++)
    {
        GOMDBTS256_readCalibrationEntryInfo(handle, i, calibInfo);
        if (*calibInfo != '\0')
        {
```

```
            std::cout « i « ":
 « calibInfo « std::endl;
        }
    }
    return 0;
}
int __stdcall BTS256Import::setCalibrationEntry(int value)
{
    int l_rc = GOMDBTS256_setCalibrationEntryNumber(handle, value);
    return l_rc;
}
int __stdcall BTS256Import::setSpectralMeasurementSettings(int integrationtime)
{
    int l_rc = GOMDBTS256_spectralSetIntegrationTimeInMs(handle, integrationtime);
    return l_rc;
}
int __stdcall BTS256Import::measure()
{
    int l_rc = GOMDBTS256_measure(handle);
    return l_rc;
}
int __stdcall BTS256Import::integralGetValues(double* value, char* unit)
{
    int l_rc = GOMDBTS256_getCWValue(handle, value);
    if (l_rc < 0)
        return l_rc;
    int calibrationEntryNumber;
    l_rc = GOMDBTS256_getSelectedCalibrationEntryNumber(handle, &calibrationEntryNumber);
    if (l_rc < 0)
        return l_rc;
    l_rc = GOMDBTS256_integralGetUnit(handle, calibrationEntryNumber, unit);
    return l_rc;
}
int __stdcall BTS256Import::close()
{
    int l_rc = GOMDBTS256_releaseHandle(handle);
    handle = -1;
    return l_rc;
}
bool __stdcall BTS256Import::getProcAddresses(HINSTANCE *p_hLibrary,
    const char* p_dllName, INT p_count, ...)
{
    va_list l_va;
    va_start(l_va, p_count);
    if ((*p_hLibrary = LoadLibrary(p_dllName)) != NULL)
    {
        FARPROC* l_procFunction = NULL;
        char* l_funcName = NULL;
        int l_idxCount = 0;
        while (l_idxCount < p_count)
        {
            l_procFunction = va_arg(l_va, FARPROC*);
            l_funcName = va_arg(l_va, LPSTR);
            if ((*l_procFunction =
                GetProcAddress(*p_hLibrary, l_funcName)) == NULL)
            {
                l_procFunction = NULL;
                return FALSE;
            }
            l_idxCount++;
        }
    }
    else
    {
        va_end(l_va);
        return false;
    }
    va_end(l_va);
    return true;
}
```

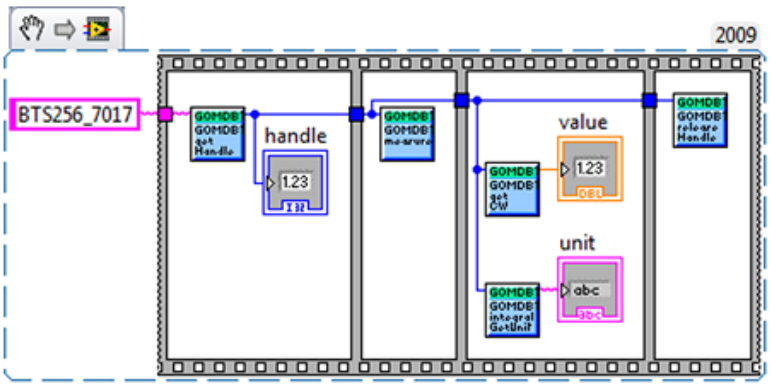
2.1.3 BTS256Import.h

```
#ifndef BTS256ImportH
#define BTS256ImportH
#include <Windows.h>
#include ßtdio.h
#include <iostream>
class BTS256Import
{
public:
    BTS256Import();
    virtual  BTS256Import();
    int __stdcall init(char* deviceName);
```

```
    int __stdcall close();
    int __stdcall writeCalibrationInfoToConsole();
    int __stdcall setCalibrationEntry(int value);
    int __stdcall setSpectralMeasurementSettings(int integrationtime);
    int __stdcall integralGetValues(double* value, char* unit);
    int __stdcall measure();
private:
    int handle;
    HINSTANCE hDLLGOBTS256;
    bool __stdcall getProcAddresses(HINSTANCE *p_hLibrary, const char* p_dllName, int p_count, ...);
    int(__stdcall *GOMDBTS256_setPassword)(char* password);
    int(__stdcall *GOMDBTS256_getHandle)(char* deviceName, int* handle);
    int(__stdcall *GOMDBTS256_releaseHandle)(int handle);
    int(__stdcall *GOMDBTS256_setCalibrationEntryNumber)(int handle, int calibrationEntryNumber);
    int(__stdcall *GOMDBTS256_getSelectedCalibrationEntryNumber)(int handle, int* calibrationEntryNumber);
    int(__stdcall *GOMDBTS256_readCalibrationEntryInfo)(int handle, int calibrationEntryNumber, char*
       calibrationName);
    int(__stdcall *GOMDBTS256_spectralSetIntegrationTimeInMs)(int handle, int timeInMs);
    int(__stdcall *GOMDBTS256_measure)(int handle);
    int(__stdcall *GOMDBTS256_getCWValue)(int handle, double* value);
    int(__stdcall *GOMDBTS256_integralGetUnit)(int handle, int calibrationEntryNumber, char* unit);
};
#endif
```

2.2 LabView Example

This example demonstrates getting a device handle which is used for every consecutive VI-call. Then a measurement is done, measurement values get collected and written to a indicator. At the end the handle will be released to free all resources.



2.3 More Examples

Further examples for integrating DLL's, can be found in the installation directory of the SDK.

Kapitel 3

Change History

A list of all modifications of the S-SDK BTS256 follows:

- **V1.0**
  Initial version
- **V2013.3**
  New: Elimination of configuration files. Configuration files are not needed any more, if complete calibration information is stored within measurement device. This procedure must be done in factory.
  Bugfixes
- **V2014.1**
  New: Methods for flicker
  New: Implementation of WIFI functionality for use with WIFI BTS256 devices
- **V2014.2**
  New: method spectralSetScaleWithVLambda
  New: method spectralIsScaleWithVLambda
  New: method spectralSetBandwidthCorrection
  New: method spectralIsBandwidthCorrection
  New: method getUserSpecificWeightingValue
- **V2014.3**
  Internal changes
- **V2014.4**
  New: method getPurity(int handle, double∗ value);
  New: method getEsy1(int handle, double∗ value);
  New: method getEsy2(int handle, double∗ value);
  New: method getEch(int handle, double∗ value);
  New: method getEmoPr(int handle, double∗ value);
  New: method getEmoPfr(int handle, double∗ value);
  New: method getEpt1(int handle, double∗ value);
  New: method getEpt2(int handle, double∗ value);
  New: method getQsy(int handle, double∗ value);
  New: method getSerialNumber(int handle, char∗ value);
  New: method getType(int handle, char∗ value);
  New: method setAreaSize(int handle, double value);
  New: method getAreaSize(int handle, double∗ value);
  New: method isCalibrationEntryVirtual(int handlem, int p_calibrationEntryNumber, char∗ quantity, bool∗ is↩ Virtual);
  New: method importUserWeightingFactors
  New: method getUserWeightingFactors
  New: method deleteUserWeightingFactors
  New: method getUserWeightingText

- **V2015.1**
  bugfix: memory overflow
- **V2016.1**
  New: password protection
  New: methods for melaopsin calculation
  New: method showConfig removed
- **V2017.1**
  Update: performance improvements
  New: 10 degree observer
  New: number of scans
  New: Flicker Measurement with 50kS/s
- **V2018.1**
  New: TLA measurement routine
  New: Values PST,SVM,Assist Mp
  New: alpha-optic values added
- **V2018.2**
  Update: Update and Bugfixes for TLA measurement
  Update: Support for new BTS256 and LED-Tester
- **V2018.3**
  New: SDK release for x64
  Update: hide flicker values for low AC
  Update: support for BTS256 Redesign
- **V2019.1**
  Update: USB Performance
  Bugfix: FFT Values: Removing duplicate entries
  Bugfix: PST: Inaccuracy for small values
- **V2019.2**
  New: TLCI Calculation
  Update: Stability of TLA measurement
- **V2019.3**
  Update: WL Range for LED Tester
- **V2019.4**
  Update: TM-30-15 now fullfilling TM-30-18
  Bugfix: Flicker Emin, Emax
- **V2019.5**
  New: JA10 of California Energy Commision
  New: Color CIE170
  New: Support BTS256-UV-1 and BTS256-UV-2
- **V2019.6**
  Bugfix: TM-30-18
- **V2019.7**
  New: methods for setting and getting integral Range
- **V2020.1**
  Update: Melanopic and AlphaOpic from CIE S026
- **V2020.2**
  Update: TLA measurement with 40 kHz
- **V2020.3**
  Bugfix: Flicker: Emin, Emax
  Bugfix: FFT Values

- **V2020.4**
  New: loggerDLIReadData
- **V2020.5**
  Update Timeout for straylight correction
- **V2020.6**
  Update: Reduced memory cosumption during TLA calculation
- **V2020.7**
  Bugfix: Precission of PST measurement
- **V2020.8** - V2020.10
  Update: Error handling on flicker calculation
- **V2020.11**
  New: Support for BTS256-UV3 and BTS256-UV4
- **V2021.1**
  Bugfix: Shortened calibrationname during recalibration
  Bugfix: Reading lampfile of calibration routine

Kapitel 4

Errors and Warnings

A list of errors and warnings follows.

4.1 Errors

- -5000: general communication problem
- -5001: setup file not valid for BTS256
- -5002: setup file could not be opened
- -5003: config file not found
- -5004: az mode out of range (0 - 2 allowed)
- -5005: communication channel can not be opened
- -5006: firmwareversion to low
- -5007: communication send problem
- -5008: communication receive problem
- -5009: device returned error
- -5010: delta uv limit $<$ 0
- -5014: error main data eeprom
- -5015: error color data eeprom
- -5017: error zero adjust integral amplifier
- -5020: error dark current measurement
- -5023: error BTS256 overload
- -5024: error user data eeprom
- -5060: wrong password
- -5997: no BTS256 connected
- -5998: device connected with different serial number
- -5999: unknown error
- -5105: invalid handle

4.2 Warnings

- 5011: underload integral sensor
- 5012: overload integral sensor
- 5025: warning BTS256 low signal
- 5075: a(Z) mode changed

Kapitel 5

Module Documentation

5.1 Information

Information about the usage of the SDK methods.

All the methods described here can be applied to every BTS256. Certain differences in the application can arise depending on the configuration, calibration and features of your measurement device. For instance, some methods may fail to provide any results for certain device configurations.
Each method provides a return value. Return value “0” means error-free execution of the method. Values less than “0” indicate the occurrence of an error. Values larger than “0” should be regarded as warnings.
A list of all return values is include in the documentation.

5.2 Standard SDK Methods

Functions

- int GOFUNCDESC GOMDBTS256_setPassword (char ∗value)
- int GOFUNCDESC GOMDBTS256_getHandle (char ∗deviceName, int ∗handle)
- int GOFUNCDESC GOMDBTS256_releaseHandle (int handle)
- int GOFUNCDESC GOMDBTS256_getDLLVersion (char ∗value)
- int GOFUNCDESC GOMDBTS256_getFirmwareVersion (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_getSerialNumber (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_getType (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_switchLight (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_disableButtons (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_getTemperature (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_switchPowerAuxLED (int handle, bool powerOn, int current)
- int GOFUNCDESC GOMDBTS256_isConnected (int handle, bool ∗value)

5.2.1 Detailed Description

Methods for handlings the SDK and Device.

5.2.2 C++ Example

This example describes the initialization of your device. This is the default structure of initialization:

```
GOMDBTS256_setPassword("Your password
");
int handle;
int l_rc = GOMDBTS256_getHandle("X1_0
", &handle);       //initialization of device
if (handle > 0 )
{
// do something
}
GOMDBTS256_releaseHandle(handle);                    //re-initialization of device
```

5.2.3 Function Documentation

5.2.3.1 GOMDBTS256_setPassword()

```
int GOFUNCDESC GOMDBTS256_setPassword (
              char * value )
```

This method has to be called before any other to activate the SDK. Activation takes place on several levels.

- 1. level: general use of the SDK
- 2. level: all elements of the 1st level plus saving of the calibrations in the customized memory The passwords are separately provided to you by Gigahertz-Optik.

Parameters

in	*value*	Zero terminated string containing the password.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.2 GOMDBTS256_getHandle()

```
int GOFUNCDESC GOMDBTS256_getHandle (
            char * deviceName,
            int * handle )
```

This method you have to call initially. You will receive an identifier, which you have to use in any further method call.

Parameters

in	*deviceName*	string value which identifies your BTS256 device. This value always consists of the prefix „BTS256_“ and appends the serial number of your BTS256 device, e.g. “BTS256_5678”. If serial is 0, then any connected BTS256 will be initialized.
out	*handle*	Pointer to integer; this value has to be used as input when calling any further BTS256 method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.3 GOMDBTS256_releaseHandle()

```
int GOFUNCDESC GOMDBTS256_releaseHandle (
            int handle )
```

This method has to be called at the end of your application to free all resources.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.4 GOMDBTS256_getDLLVersion()

```
int GOFUNCDESC GOMDBTS256_getDLLVersion (
            char * value )
```

Returns the version number of this DLL.

Parameters

out	*value*	Null-terminated string; contains the version number after return, minimum size: 10 bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.5 GOMDBTS256_getFirmwareVersion()

```
int GOFUNCDESC GOMDBTS256_getFirmwareVersion (
            int handle,
            char * value )
```

Returns the firmware version of the connected BTS256.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Null-terminated string; returns the firmware version, minimum size: 10 Bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.6 GOMDBTS256_getSerialNumber()

```
int GOFUNCDESC GOMDBTS256_getSerialNumber (
            int handle,
            char * value )
```

Returns the serial number of the connected BTS256.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Null-terminated string; contains the serial number of the BTS256 after return, minimum size: 10 bytes

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.7 GOMDBTS256_getType()

```
int GOFUNCDESC GOMDBTS256_getType (
            int handle,
            char * value )
```

Returns the device type of the connected BTS256.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Null-terminated string; contains the type of the BTS56 after return, minimum size: 30 bytes

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.8 GOMDBTS256_switchLight()

```
int GOFUNCDESC GOMDBTS256_switchLight (
            int handle,
            bool value )
```

Attention

Only for BTS256-E

Switches display light on/ off.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Switch light on • false: Switch light off

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.9 GOMDBTS256_disableButtons()

```
int GOFUNCDESC GOMDBTS256_disableButtons (
            int handle,
            bool value )
```

Attention

Only for BTS256-E

This method disables / enables the device buttons.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Disable buttons • false: Enable buttons

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.10 GOMDBTS256_getTemperature()

```
int GOFUNCDESC GOMDBTS256_getTemperature (
            int handle,
            double * value )
```

Attention

Only for BTS256-E

This method returns the temperature of the internal temperature sensor in [°C].

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the temperature in [°C].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.11 GOMDBTS256_switchPowerAuxLED()

```
int GOFUNCDESC GOMDBTS256_switchPowerAuxLED (
            int handle,
            bool powerOn,
            int current )
```

Switches internal auxiliary lamp on/off, which is needed when substitution correction measurements should be performed for the internal sphere of the device. When using an external sphere you have to switch on the auxiliary with the corresponding commands of the SDK for your power supply. Normally you don't need to use this method, because the LED will be switched on/off automatically when calling "substitutionMeasurementWithoutTestDevice" or "substitutionMeasurementWithTestDevice"

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*powerOn*	Boolean value, true to switch on, false to switch off.
in	*current*	Integer value, current in [mA], a value of 25mA is good for internal.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.2.3.12 GOMDBTS256_isConnected()

```
int GOFUNCDESC GOMDBTS256_isConnected (
            int handle,
            bool * value )
```

This method checks if the device is still connected to the PC or has been disconnected

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
`out`	*value*	Pointer to boolean; gives the connection-status:

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3 Methods of common measurement settings

Functions

- int GOFUNCDESC GOMDBTS256_saveConfig (int handle, char ∗filename)
- int GOFUNCDESC GOMDBTS256_loadConfig (int handle, char ∗filename)
- int GOFUNCDESC GOMDBTS256_saveConfigAsDefault (int handle)
- int GOFUNCDESC GOMDBTS256_getCWValue (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_setCalibrationEntryNumber (int handle, int calibrationEntryNumber)
- int GOFUNCDESC GOMDBTS256_getSelectedCalibrationEntryNumber (int handle, int ∗calibrationEntry↩ Number)
- int GOFUNCDESC GOMDBTS256_readCalibrationEntryInfo (int handle, int calibrationEntryNumber, char ∗calibrationName)
- int GOFUNCDESC GOMDBTS256_getMeasurementQuantity (int handle, int calibrationEntryNumber, char ∗quantity)
- int GOFUNCDESC GOMDBTS256_isMeasurementQuantity (int handle, int calibrationEntryNumber, char ∗quantity, bool ∗isQuantity)
- int GOFUNCDESC GOMDBTS256_getSelectedMeasurementQuantity (int handle, char ∗quantity)
- int GOFUNCDESC GOMDBTS256_setDistance (int handle, double distance)
- int GOFUNCDESC GOMDBTS256_getDistance (int handle, double ∗distance)
- int GOFUNCDESC GOMDBTS256_setDeltaUVLimit (int handle, double limit)
- int GOFUNCDESC GOMDBTS256_getDeltaUVLimit (int handle, double ∗limit)
- int GOFUNCDESC GOMDBTS256_setAreaSize (int handle, double value)
- int GOFUNCDESC GOMDBTS256_getAreaSize (int handle, double ∗value)

5.3.1 Detailed Description

Common Measurement settings for the BTS256.

5.3.2 C++ Example

This example describes the initialization of your device. This is the default structure of initialization:

```
GOMDBTS256_setPassword("Your password
");
int handle;
int l_rc = GOMDBTS256_getHandle("X1_0
", &handle);       //initialization of device
if (handle > 0 )
{
// do something
}
GOMDBTS256_releaseHandle(handle);                       //re-initialization of device
```

5.3.3 Function Documentation

5.3.3.1 GOMDBTS256_saveConfig()

```
int GOFUNCDESC GOMDBTS256_saveConfig (
              int handle,
              char * filename )
```

The currently set parameters are saved in a configuration file for later use. The values can be loaded using “load↩ Config”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*filename*	zero terminated string; file name including path where the configuration data should be saved

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.2 GOMDBTS256_loadConfig()

```
int GOFUNCDESC GOMDBTS256_loadConfig (
            int handle,
            char * filename )
```

This method loads all previously set and saved values from the specified file (see the mehtod saveConfig). If the configuration file does not belong to a BTS256 but rather to a different device, an error code is returned.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*filename*	Complete path to a configuration file where pre-existing settings are saved.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.3 GOMDBTS256_saveConfigAsDefault()

```
int GOFUNCDESC GOMDBTS256_saveConfigAsDefault (
            int handle )
```

The currently set parameters are saved in a configuration file for later use and are reloaded upon re-initialization of the BTS256.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.4 GOMDBTS256_getCWValue()

```
int GOFUNCDESC GOMDBTS256_getCWValue (
            int handle,
            double * value )
```

Returns calculated (calibration value, correction, ...) measurement value of integral device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double, measurement value of integral device.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.5 GOMDBTS256_setCalibrationEntryNumber()

```
int GOFUNCDESC GOMDBTS256_setCalibrationEntryNumber (
            int handle,
            int calibrationEntryNumber )
```

Your device is delivered with one or more calibrations. These are ideal for different measurement scenarios. This method enables you to select calibrations saved in EEPROM. A total of 20 calibration entries exist and not all have to be filled. To find all valid calibrations, the method readCalibrationEntryInfo can bew used. If a non-existent calibration index is selected, the method returns an error code.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.6 GOMDBTS256_getSelectedCalibrationEntryNumber()

```
int GOFUNCDESC GOMDBTS256_getSelectedCalibrationEntryNumber (
            int handle,
            int * calibrationEntryNumber )
```

Returns actual selected calibration entry number.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.7 GOMDBTS256_readCalibrationEntryInfo()

```
int GOFUNCDESC GOMDBTS256_readCalibrationEntryInfo (
            int handle,
            int calibrationEntryNumber,
            char * calibrationName )
```

Returns the name of the specified calibration entry. The calibration name is defined in the device configuration file BTS256_<serialnumber>.goi, which is delivered with your system. Allocate enough memory for calibrationName that the name can be copied into.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.
out	*calibrationName*	string value, contains the name of the calibration

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.8 GOMDBTS256_getMeasurementQuantity()

```
int GOFUNCDESC GOMDBTS256_getMeasurementQuantity (
            int handle,
            int calibrationEntryNumber,
            char * quantity )
```

Returns the quantity for the specified calibration entry number. Possible return values are “E”, “I” or “Phi”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.
out	*quantity*	String value, “E”, “I”, “Phi”.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.9 GOMDBTS256_isMeasurementQuantity()

```
int GOFUNCDESC GOMDBTS256_isMeasurementQuantity (
            int handle,
            int calibrationEntryNumber,
            char * quantity,
            bool * isQuantity )
```

Returns if the specified calibration entry is defined for a special quantity “E”, “I”, “Phi”. You can have a look in the device configuration file to see the defined calibration entries and their assigned quantities.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Parameters

in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.
in	*quantity*	string value, “E”, “I”, “Phi”.
out	*isQuantity*	Pointer to boolean, true if specified calibration entry is assigned for specified quantity

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.10 GOMDBTS256_getSelectedMeasurementQuantity()

```
int GOFUNCDESC GOMDBTS256_getSelectedMeasurementQuantity (
            int handle,
            char * quantity )
```

Returns the quantity of the actual selected calibration entry. Possible values are “E”, “I” or “Phi”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*quantity*	String value, possible values: “E”, “I”, “Phi”.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.11 GOMDBTS256_setDistance()

```
int GOFUNCDESC GOMDBTS256_setDistance (
            int handle,
            double distance )
```

If you selected a calibration entry for “Luminous Intensity” or “Radiant Intensity, then you must define the distance between your BTS256 measurement device and your “device under test”. Distance unit is [m].

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*distance*	Double value, the distance between BTS256 and device under test in [m].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.12 GOMDBTS256_getDistance()

```
int GOFUNCDESC GOMDBTS256_getDistance (
            int handle,
            double * distance )
```

Returns the previous set distance between BTS256 and “device under test” in [m].

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*distance*	Pointer to double value, the previous defined distance between BTS256 and “device under test”.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.13 GOMDBTS256_setDeltaUVLimit()

```
int GOFUNCDESC GOMDBTS256_setDeltaUVLimit (
            int handle,
            double limit )
```

Defines the limit you want to set for calculation of CCT. If real values are outside limit, then CCT will not be calculated. Default value is “0.05”. This method/VI needs firmware versions >= “0.43”.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*limit*	Double value, the limit you want to set for calculation of CCT.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.14 GOMDBTS256_getDeltaUVLimit()

```
int GOFUNCDESC GOMDBTS256_getDeltaUVLimit (
            int handle,
            double * limit )
```

Returns the previous set uv limit for calculation of CCT.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*limit*	Pointer to double value, the limit for calculation of CCT

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.15 GOMDBTS256_setAreaSize()

```
int GOFUNCDESC GOMDBTS256_setAreaSize (
            int handle,
            double value )
```

If you selected a calibration entry for "Luminance" or "Radiance", then you must define the area size of your "device under test". The unit is [m^2].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Double value, the area size device under test in [m^2].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.3.3.16 GOMDBTS256_getAreaSize()

```
int GOFUNCDESC GOMDBTS256_getAreaSize (
            int handle,
            double * value )
```

Returns the previous set area size of your “device under test” in [m^2].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, the previous defined area size of your “device under test”.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4 Methods of integral measurement settings

Functions

- int GOFUNCDESC GOMDBTS256_integralSetEnabled (int handle, bool enabled)
- int GOFUNCDESC GOMDBTS256_integralIsEnabled (int handle, bool ∗enabled)
- int GOFUNCDESC GOMDBTS256_integralSetIntegrationTimeInMs (int handle, int timeInMs)
- int GOFUNCDESC GOMDBTS256_integralGetIntegrationTimeInMs (int handle, int ∗timeInMs)
- int GOFUNCDESC GOMDBTS256_integralSetSynchronization (int handle, int value)
- int GOFUNCDESC GOMDBTS256_integralGetSynchronization (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_integralSetSynchronizationTimeInMs (int handle, int valueInMs)
- int GOFUNCDESC GOMDBTS256_integralGetSynchronizationTimeInMs (int handle, int ∗valueInMs)
- int GOFUNCDESC GOMDBTS256_integralSetRange (int handle, int value)
- int GOFUNCDESC GOMDBTS256_integralGetRange (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_integralGetLastUsedRange (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_integralGetUnit (int handle, int calibrationEntryNumber, char ∗unit)
- int GOFUNCDESC GOMDBTS256_integralGetLastUsedAz (int handle, double ∗az)
- int GOFUNCDESC GOMDBTS256_integralSetAzSpecific (int handle, double az)
- int GOFUNCDESC GOMDBTS256_integralGetAzSpecific (int handle, double ∗az)
- int GOFUNCDESC GOMDBTS256_integralSetAzMode (int handle, int option)
- int GOFUNCDESC GOMDBTS256_integralGetAzMode (int handle, int ∗mode)

5.4.1 Detailed Description

Measurement settings for the integral measurement.

5.4.2 Function Documentation

5.4.2.1 GOMDBTS256_integralSetEnabled()

```
int GOFUNCDESC GOMDBTS256_integralSetEnabled (
            int handle,
            bool enabled )
```

This activates/deactivates integral sensor for measurement. If deactivated, integral sensor won't be measured.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*enabled*	Boolean value, true to activate sensor, false to deactivate sensor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.2 GOMDBTS256_integralIsEnabled()

```
int GOFUNCDESC GOMDBTS256_integralIsEnabled (
            int handle,
            bool * enabled )
```

Checks if integral sensor is activated for measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*enabled*	Pointer to boolean value: • true if active • false if not active.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.3 GOMDBTS256_integralSetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_integralSetIntegrationTimeInMs (
            int handle,
            int timeInMs )
```

This sets the integration time in milliseconds of the integral sensor to be measured. Minimum allowed time is 1ms and maximum time is 6000ms.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*timeInMs*	Pointer to integer, the time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.4 GOMDBTS256_integralGetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_integralGetIntegrationTimeInMs (
            int handle,
            int * timeInMs )
```

This method returns the integration time for the integral sensor in milliseconds.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Pointer to integer, receives the time in milliseconds which was set before.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.5 GOMDBTS256_integralSetSynchronization()

```
int GOFUNCDESC GOMDBTS256_integralSetSynchronization (
              int handle,
              int value )
```

Defines if measurement should be synchronized to your signal, if possible.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Integer value, 1 to activate sync., 0 to deactivate sync.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.6 GOMDBTS256_integralGetSynchronization()

```
int GOFUNCDESC GOMDBTS256_integralGetSynchronization (
              int handle,
              int * value )
```

Returns if synchronization to your signal is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to integer value: • 1: If sync. is active • 0: If sync. is not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.7 GOMDBTS256_integralSetSynchronizationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_integralSetSynchronizationTimeInMs (
            int handle,
            int valueInMs )
```

Sets the time in [ms], the measurement device examines the signal to find loops.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*valueInMs*	integer value; the synchronization time in [ms].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.8 GOMDBTS256_integralGetSynchronizationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_integralGetSynchronizationTimeInMs (
            int handle,
            int * valueInMs )
```

Returns the previous set synchronization time in [ms].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*valueInMs*	Pointer to integer value; the synchronization time in [ms].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.9 GOMDBTS256_integralSetRange()

```
int GOFUNCDESC GOMDBTS256_integralSetRange (
            int handle,
            int value )
```

The integral range defines the sensitivity of the integral detector. There are 9 ranges available: (0 = insensitive, 8 = sensitive) and an autorange option (-1). If you picked up the wrong range the measurement will lead to an "integral overload" or "integral underload".
In autorange mode, the device independently searches for the optimum measurement range for the signal to be measured.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	integer value; contains the measurement range: • -1: Auto ranging • 0 – 8: Specific measurement range

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.10 GOMDBTS256_integralGetRange()

```
int GOFUNCDESC GOMDBTS256_integralGetRange (
            int handle,
            int * value )
```

This methods reads the setting of the integral range that will operate on the next measurement. If you want to read out the integral range of the last measurement you have to use the method integralGetLastUsedRange().

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	pointer to integer value; contains the measurement range: • -1: Auto ranging • 0 – 8: Specific measurement range

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.11 GOMDBTS256_integralGetLastUsedRange()

```
int GOFUNCDESC GOMDBTS256_integralGetLastUsedRange (
            int handle,
            int * value )
```

This methods reads the integral range of the last measurement performed. If you have set the integarl range to auto, this methods will give you the range that was actual used during the measurment.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	pointer to integer value; contains the measurement range used during last measurment: (0-8)

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.12 GOMDBTS256_integralGetUnit()

```
int GOFUNCDESC GOMDBTS256_integralGetUnit (
            int handle,
            int calibrationEntryNumber,
            char * unit )
```

Returns the unit, valid for integral measurements with a specified calibration entry.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.
out	*unit*	String value, contains the unit for the specified calibration entry number.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.13 GOMDBTS256_integralGetLastUsedAz()

```
int GOFUNCDESC GOMDBTS256_integralGetLastUsedAz (
            int handle,
            double * az )
```

Returns the a(z) value used for last measurement to correct the integral sensor value. This may be a calculated value from the measured spectrum or can be a static defined value or 1 if no a(z) correction should be done.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*az*	Pointer to double value, contains the last used az value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.14 GOMDBTS256_integralSetAzSpecific()

```
int GOFUNCDESC GOMDBTS256_integralSetAzSpecific (
            int handle,
            double az )
```

Defines a static value for a(z) correction of integral measurement. To activate this value you also have to define the a(z) mode to be static.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*az*	Double value, contains the static a(z) correction factor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.15 GOMDBTS256_integralGetAzSpecific()

```
int GOFUNCDESC GOMDBTS256_integralGetAzSpecific (
            int handle,
            double * az )
```

Returns the previous set static a(z) correction factor which will be used if static a(z)-mode is active.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*az*	Pointer to double value, contains the previous defined static a(z) correction factor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.16 GOMDBTS256_integralSetAzMode()

```
int GOFUNCDESC GOMDBTS256_integralSetAzMode (
            int handle,
            int option )
```

Defines the correction mode for the integral sensor.

- mode = 0: no correction
- mode = 1: correction with static factor set by call of “integralSetAzSpecific”
- mode = 2: correction with dynamic calculated a(z) factor, array measurement must be enabled.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*option*	Integer value, the az mode that should be used for correction.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.4.2.17 GOMDBTS256_integralGetAzMode()

```
int GOFUNCDESC GOMDBTS256_integralGetAzMode (
            int handle,
            int * mode )
```

Returns the previous defined correction mode for the integral sensor.

- mode = 0: no correction
- mode = 1: correction with static factor set by call of “integralSetAzSpecific”
- mode = 2: correction with dynamic calculated a(z) factor, array measurement must be enabled.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*mode*	Pointer to integer value, contains a(z) correction mode.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5 Methods of spectral measurement settings

Functions

- int GOFUNCDESC GOMDBTS256_spectralSetEnabled (int handle, bool enabled)
- int GOFUNCDESC GOMDBTS256_spectralIsEnabled (int handle, bool ∗enabled)
- int GOFUNCDESC GOMDBTS256_setWavelengthRange (int handle, double L1, double L2, double dL)
- int GOFUNCDESC GOMDBTS256_getWavelengthRange (int handle, double ∗L1, double ∗L2, double ∗dL)
- int GOFUNCDESC GOMDBTS256_spectralSetIntegrationTimeInMs (int handle, int timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralGetIntegrationTimeInMs (int handle, int ∗timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralSetIntegrationTimeMaxInMs (int handle, int timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralGetIntegrationTimeMaxInMs (int handle, int ∗timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralGetNrOfScans (int handle, int ∗timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralSetNrOfScans (int handle, int value)
- int GOFUNCDESC GOMDBTS256_getObserver10degree (int handle, bool ∗value)
- int GOFUNCDESC GOMDBTS256_spectralSetSignalResolution (int handle, int resolution)
- int GOFUNCDESC GOMDBTS256_spectralGetSignalResolution (int handle, int ∗resolution)
- int GOFUNCDESC GOMDBTS256_spectralSetOffsetMode (int handle, int value)
- int GOFUNCDESC GOMDBTS256_spectralGetOffsetMode (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_spectralSetDynamicTimeMode (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_spectralGetDynamicTimeMode (int handle, bool ∗value)
- int GOFUNCDESC GOMDBTS256_getMinValidWavelength (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getMaxValidWavelength (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getMinValidStepwidth (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_setObserver10degree (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_spectralSetDarkThreshold (int handle, int value)
- int GOFUNCDESC GOMDBTS256_spectralGetDarkThreshold (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_spectralSetScaleWithDiode (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_spectralIsScaleWithDiode (int handle, bool ∗value)
- int GOFUNCDESC GOMDBTS256_spectralSetBandwidthCorrection (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_spectralIsBandwidthCorrection (int handle, bool ∗value)

5.5.1 Detailed Description

Measurement settings for the spectral measuremt.

5.5.2 Function Documentation

5.5.2.1 GOMDBTS256_spectralSetEnabled()

```
int GOFUNCDESC GOMDBTS256_spectralSetEnabled (
            int handle,
            bool enabled )
```

Activates / deactivates array for measurement. If array is deactivated, no a(z) correction factor is calculated. This means you should set a static a(z) value for integral correction.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*enabled*	boolean value, • true to activate array • false to deactivate array

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.2 GOMDBTS256_spectralIsEnabled()

```
int GOFUNCDESC GOMDBTS256_spectralIsEnabled (
            int handle,
            bool * enabled )
```

Returns, if spectral unit is active or not active for measurements.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*enabled*	Pointer to boolean: • true if active • false if not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.3 GOMDBTS256_setWavelengthRange()

```
int GOFUNCDESC GOMDBTS256_setWavelengthRange (
            int handle,
            double L1,
            double L2,
            double dL )
```

Defines the limiting factor used when “spectralGetCountsWavelength” or “spectralGetSpectrumCalibrated↩ Wavelength” is called.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*L1*	Double value, minimum wavelength in [nm]
in	*L2*	Double value, maximum wavelength in [nm]
in	*dL*	Double value, stepwidth in [nm]

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.4 GOMDBTS256_getWavelengthRange()

```
int GOFUNCDESC GOMDBTS256_getWavelengthRange (
            int handle,
            double * L1,
            double * L2,
            double * dL )
```

Returns the previous defined wavelength range, which will be used when “spectralGetCountsWavelength” or “spectralGetSpectrumCalibratedWavelength” is called.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*L1*	Pointer to double, minimum wavelength in [nm]
out	*L2*	Pointer to double, maximum wavelength in [nm]
out	*dL*	Pointer to double, stepwidth in [nm]

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.5 GOMDBTS256_spectralSetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_spectralSetIntegrationTimeInMs (
            int handle,
            int timeInMs )
```

Defines the integration time in milliseconds, which will be used for spectral measurement. If set to “0” then BTS256 device will calculate the best possible integration time dependent from your signal.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Integer value, the integration time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.6 GOMDBTS256_spectralGetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_spectralGetIntegrationTimeInMs (
             int handle,
             int * timeInMs )
```

Returns the real integration time in milliseconds used for your spectral device during last measurement.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Pointer to integer; contains the integration time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.7 GOMDBTS256_spectralSetIntegrationTimeMaxInMs()

```
int GOFUNCDESC GOMDBTS256_spectralSetIntegrationTimeMaxInMs (
             int handle,
             int timeInMs )
```

If spectral integration time is set to dynamic (integration time = 0ms), then this method limits the dynamic time to a maximum time that will be used for spectral measurement. This method mustn't be called after setting the spectral integration time to a specific value >= 5ms. Every time, the method "spectralSetIntegrationTimeInMs" is called, the maximum integration time is reset to 30000ms.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*timeInMs*	Integer value, the integration time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.8 GOMDBTS256_spectralGetIntegrationTimeMaxInMs()

```
int GOFUNCDESC GOMDBTS256_spectralGetIntegrationTimeMaxInMs (
            int handle,
            int * timeInMs )
```

Returns the actual set maximum integration time value for dynamic integration time evaluation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Pointer to integer; contains the integration time in milliseconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.9 GOMDBTS256_spectralGetNrOfScans()

```
int GOFUNCDESC GOMDBTS256_spectralGetNrOfScans (
            int handle,
            int * timeInMs )
```

Returns the actual set maximum integration time value for dynamic integration time evaluation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Pointer to integer; contains the integration time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.10 GOMDBTS256_spectralSetNrOfScans()

```
int GOFUNCDESC GOMDBTS256_spectralSetNrOfScans (
            int handle,
            int value )
```

This method sets the number of scans that should be performed at each measurement. The result value will be a mean of all measurements.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	integer value containing the number of scans.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.11 GOMDBTS256_getObserver10degree()

```
int GOFUNCDESC GOMDBTS256_getObserver10degree (
            int handle,
            bool * value )
```

This method returns the number of scans that was set by the method spectralSetNrOfScans.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to integer value, that contains the number of scans after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.12 GOMDBTS256_spectralSetSignalResolution()

```
int GOFUNCDESC GOMDBTS256_spectralSetSignalResolution (
            int handle,
            int resolution )
```

Defines the counts level that will be used for evaluation of dynamic integration time for your spectral device. Possible values:

- 0: low
- 1: middle
- 2: high
  "High" means that the integration time will be longer to get more counts. "Low" means, that even fewer counts will be accepted as okay for dynamic integration time evaluation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*resolution*	Integer value: • 0: low • 1: middle • 2: high

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.13 GOMDBTS256_spectralGetSignalResolution()

```
int GOFUNCDESC GOMDBTS256_spectralGetSignalResolution (
            int handle,
            int * resolution )
```

Returns the previous set level, that will be used for dynamic integration time evaluation of your spectral device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*resolution*	Pointer to integer value: • 0: low • 1: middle • 2: high

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.14 GOMDBTS256_spectralSetOffsetMode()

```
int GOFUNCDESC GOMDBTS256_spectralSetOffsetMode (
            int handle,
            int value )
```

This method defines the offset mode used for the measurement. There are three different modes: no offset, static and dynamic.
No offset means that no offset is subtracted from the measured signal.
Static means that the offset has to be explicitly measured once. This is done using the “spectralMeasureOffset” method. Afterwards, the measured offset is used for the measurements that follow. The “spectralDeleteOffset” method is used to reset the offset back to 0. In the case of the static offset, it should be noted that the spectral integration time can no longer be changed after the offset measurement as the offset always changes with the integration time. After changing the spectral integration time, the offset has to be measured again.
In dynamic, the offset has to be determined for each measurement. The “dark filter” is hereby used automatically and the offset measured. For the actual measurement of the wanted signal, the filter is set to the previously specified position. The filter can either be explicitly set using the “setFilterPosition” method or the filter corresponding to the selected calibration entry used.
When “no offset” or “dynamic offset” is set using this method, the “spectralDeleteOffset” is then executed automatically and the last measured offset deleted.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Integer value: • 0: No offset • 1: Static offset • 2: Dynamic offset

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.15 GOMDBTS256_spectralGetOffsetMode()

```
int GOFUNCDESC GOMDBTS256_spectralGetOffsetMode (
            int handle,
            int * value )
```

This method determines the previously set offset mode.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the previously defined offset mode: • 0: No offset • 1: Static offset • 2: Dynamic offset

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.16 GOMDBTS256_spectralSetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_spectralSetDynamicTimeMode (
            int handle,
            bool value )
```

this method specifies whether the integration time for the spectral measurement unit should be dynamically determined for each measurement. If activated, the device performs a pre-measurement before the actual measurement. The actual integration time used can be retrieved after the measurement using “spectralGetIntegrationTimeInus”. Dynamic determination of the integration time is not compatible with the “static dark mode”. If the dynamic mode is activated, the dark mode is automatically changed to “dynamic”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: Dynamic integration time determination deactivated • true: Dynamic integration time determination activated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.17 GOMDBTS256_spectralGetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_spectralGetDynamicTimeMode (
            int handle,
            bool * value )
```

This method checks whether the integration time for the spectral measurement unit should be dynamically determined for every measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to boolean: • false: Dynamic integration time determination deactivated • true: Dynamic integration time determination activated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.18 GOMDBTS256_getMinValidWavelength()

```
int GOFUNCDESC GOMDBTS256_getMinValidWavelength (
            int handle,
            double * value )
```

This method returns the min. valid wave lengh of the current selcted calibration entry.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the min. valid wave lengh [nm]

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.19 GOMDBTS256_getMaxValidWavelength()

```
int GOFUNCDESC GOMDBTS256_getMaxValidWavelength (
            int handle,
            double * value )
```

This method returns the max. valid wave lengh of the current selcted calibration entry.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the max. valid wave lengh [nm]

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.20 GOMDBTS256_getMinValidStepwidth()

```
int GOFUNCDESC GOMDBTS256_getMinValidStepwidth (
            int handle,
            double * value )
```

This method returns the min. valid stepwidth of the current selcted calibration entry.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the min. valid steplength [nm]

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.21 GOMDBTS256_setObserver10degree()

```
int GOFUNCDESC GOMDBTS256_setObserver10degree (
            int handle,
            bool value )
```

This method defines the CIE Observer for the color calculation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: CIE 1931 observer with 2° range of vision • true: CIE 1964 observer with 10° range of vision

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.22 GOMDBTS256_spectralSetDarkThreshold()

```
int GOFUNCDESC GOMDBTS256_spectralSetDarkThreshold (
            int handle,
            int value )
```

The dark threshold defines the minimum number of counts required for the signal to be analyzed and processed at the respective pixel.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Integer value; number of counts that must be exceeded as the dark threshold.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.23 GOMDBTS256_spectralGetDarkThreshold()

```
int GOFUNCDESC GOMDBTS256_spectralGetDarkThreshold (
            int handle,
            int * value )
```

Returns the number of counts defined as the dark threshold.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value; number of counts defined as the dark threshold.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.24 GOMDBTS256_spectralSetScaleWithDiode()

```
int GOFUNCDESC GOMDBTS256_spectralSetScaleWithDiode (
            int handle,
            bool value )
```

This method is used to activate/deactivate the scaling mode of the spectral funktion. If scaling is activated the spectral funktion gets scaled so the radiometric value matches with the value of integral detector.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: on • false: off

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.25 GOMDBTS256_spectralIsScaleWithDiode()

```
int GOFUNCDESC GOMDBTS256_spectralIsScaleWithDiode (
            int handle,
            bool * value )
```

Returns the actual scaling mode of the spectral funktion

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	pointer to Boolean value: • true: on • false: off

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.26 GOMDBTS256_spectralSetBandwidthCorrection()

```
int GOFUNCDESC GOMDBTS256_spectralSetBandwidthCorrection (
            int handle,
            bool value )
```

this method activates and deactivates bandwidth correction. The bandwidth correction is based on a fitting method from Woolliams which is recommended by CIE TC2.51.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Correction activated • false: Correction deactivated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.5.2.27 GOMDBTS256_spectralIsBandwidthCorrection()

```
int GOFUNCDESC GOMDBTS256_spectralIsBandwidthCorrection (
            int handle,
            bool * value )
```

Checks whether bandwidth correction is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	pointer to a Boolean value: • true: Correction activated • false: Correction deactivated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6 Measurement Methods

Functions

- int GOFUNCDESC GOMDBTS256_measure (int handle)
- int GOFUNCDESC GOMDBTS256_spectralEvaluateIntegrationTimeInMs (int handle, int ∗timeInMs)
- int GOFUNCDESC GOMDBTS256_spectralMeasureDarkOffset (int handle)
- int GOFUNCDESC GOMDBTS256_spectralDeleteDarkOffset (int handle)
- int GOFUNCDESC GOMDBTS256_measureTLA (int handle, int tactNumber, double durationInMs)
- int GOFUNCDESC GOMDBTS256_dequeueTLA (int handle, int ∗nrOfValues, double ∗integralProgression↩ Values)

5.6.1 Detailed Description

Methods to prepare and perform a measurement.

5.6.2 Function Documentation

5.6.2.1 GOMDBTS256_measure()

```
int GOFUNCDESC GOMDBTS256_measure (
              int handle )
```

Performs measurement with previous set configuration settings (integration time, ...).

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6.2.2 GOMDBTS256_spectralEvaluateIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_spectralEvaluateIntegrationTimeInMs (
              int handle,
              int * timeInMs )
```

Performs a short measurement and returns the integration time that will be used for your spectral device, if dynamic evaluation of integration time is switched on (look at: spectralSetIntegrationTimeInMs). If dynamic evaluation of integration time is switched off, then your predefined integration time will be returned.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timeInMs*	Pointer to integer value; the estimated integration time in milliseconds.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6.2.3 GOMDBTS256_spectralMeasureDarkOffset()

```
int GOFUNCDESC GOMDBTS256_spectralMeasureDarkOffset (
            int handle )
```

This method is used to measure the dark offset. The shutter will be closed and the currently set integration time are hereby used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6.2.4 GOMDBTS256_spectralDeleteDarkOffset()

```
int GOFUNCDESC GOMDBTS256_spectralDeleteDarkOffset (
            int handle )
```

This method is used to delete the last measured dark offset.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6.2.5 GOMDBTS256_measureTLA()

```
int GOFUNCDESC GOMDBTS256_measureTLA (
            int handle,
            int tactNumber,
            double durationInMs )
```

Performs a TLA(temporal light artefacts)-measurement. With this method the time behaviour if a light source can be measured. It first performs a spectral measurement with previous set configuration. Then the diode records the progression of the light signal with the parameters given with the function call.
This method should be used, to record and calculate TLA values, especially PST, SVM and Assists Mp. After the measurement, the values can be collected with the method getTLA();

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*tactNumber*	integer value - this defines the sampling frequency: • -9: 40 kHz (since FW 1.78) • -3: 200 kHz • -2: 100 kHz • 0: 25 kHz • 1: 10 kHz • 2: 5 kHz • 3: 2 kHz • 4: 1 kHz
in	*durationInMs*	double value with the duration of the measurement in ms

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.6.2.6 GOMDBTS256_dequeueTLA()

```
int GOFUNCDESC GOMDBTS256_dequeueTLA (
            int handle,
            int * nrOfValues,
            double * integralProgressionValues )
```

This method is used to collect the progressions values of the TLA measurement while the measurement is performed. Call this method in a loop to get the whole signal of the light source. With this method each value can be collected only once.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
`in,out`	*nrOfValues*	pointer to integer value, defines the number of progresiion values that should be returned. after return it contains the number of values that actually has been collected.
`in`	*integralProgressionValues*	array of double values, that contain the progression after return. Has to have the size of nrOfValues.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7 Methods of measurement values

Functions

- int GOFUNCDESC GOMDBTS256_spectralGetSpectrumCalibratedWavelength (int handle, double ∗spectrum)
- int GOFUNCDESC GOMDBTS256_spectralGetCountsPixel (int handle, double ∗counts)
- int GOFUNCDESC GOMDBTS256_spectralGetCountsWavelength (int handle, double ∗counts)
- int GOFUNCDESC GOMDBTS256_spectralGetSpecmax (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_spectralGetUnit (int handle, int calibrationEntryNumber, char ∗unit)
- int GOFUNCDESC GOMDBTS256_getColor (int handle, double ∗UpperX, double ∗UpperY, double ∗UpperZ, double ∗x, double ∗y, double ∗us, double ∗vs, double ∗CCT, double ∗domWL)
- int GOFUNCDESC GOMDBTS256_getPeak (int handle, double ∗lambda, double ∗power)
- int GOFUNCDESC GOMDBTS256_getFWHM (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getCenterWavelength (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getCentroidWavelength (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getScotopic (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getScotopicPhotopic (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getEVE (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getFlicker (int handle, double ∗Emax, double ∗Emin, double ∗percent, double ∗index, double ∗frequency)
- int GOFUNCDESC GOMDBTS256_getFFT (int handle, double ∗frequencies, double ∗percentages, double ∗resolution, double ∗frequencyMax)
- int GOFUNCDESC GOMDBTS256_getJA10 (int handle, double ∗values)
- int GOFUNCDESC GOMDBTS256_getTLA (int handle, double ∗PST, double ∗SVM)
- int GOFUNCDESC GOMDBTS256_getAssistFlickerPerception (int handle, double ∗Mp)
- int GOFUNCDESC GOMDBTS256_getDeltaUV (int handle, double ∗uv)
- int GOFUNCDESC GOMDBTS256_getRadiometricValueOverWLRange (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getUVValues (int handle, double ∗UV_A, double ∗UV_B, double ∗UV_C)
- int GOFUNCDESC GOMDBTS256_getCRI (int handle, double ∗Ra, double ∗R1, double ∗R2, double ∗R3, double ∗R4, double ∗R5, double ∗R6, double ∗R7, double ∗R8, double ∗R9, double ∗R10, double ∗R11, double ∗R12, double ∗R13, double ∗R14)
- int GOFUNCDESC GOMDBTS256_getCRI2 (int handle, double ∗Ra, double ∗R1, double ∗R2, double ∗R3, double ∗R4, double ∗R5, double ∗R6, double ∗R7, double ∗R8, double ∗R9, double ∗R10, double ∗R11, double ∗R12, double ∗R13, double ∗R14, double ∗R15)
- int GOFUNCDESC GOMDBTS256_integralProgressionGetNrOfValues (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_integralProgressionGetCalibratedValues (int handle, double ∗values)
- int GOFUNCDESC GOMDBTS256_getBLH (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_getMelanopic (int handle, double ∗Ev, double ∗Eez, double ∗Evmel)
- int GOFUNCDESC GOMDBTS256_getBilirubin (int handle, double ∗bilirubinIEC, double ∗bilirubinAAP)
- int GOFUNCDESC GOMDBTS256_getColorCIE170 (int handle, double ∗L_MB, double ∗M_MB, double ∗S↩ _MB, double ∗X_F, double ∗Y_F, double ∗Z_F)

5.7.1 Detailed Description

Methods to get measurement values.

5.7.2 Function Documentation

5.7.2.1 GOMDBTS256_spectralGetSpectrumCalibratedWavelength()

```
int GOFUNCDESC GOMDBTS256_spectralGetSpectrumCalibratedWavelength (
              int handle,
              double * spectrum )
```

Returns the calculated values with calibration factors, substitution factor, and so on of your spectral unit. Number of values should be evaluated with “spectralGetSpecmax”, that preallocated array is big enough to hold all expected values. Labview users don’t have to care about the number of values here.

- 1st element contains value for first wavelength interpolation point + 0 ∗ stepwidth
- 2nd element contains value for first wavelength interpolation point + 1 ∗ stepwidth
- ...
  The wavelength interpolation points are defined by call of “setWavelengthRange”, where you can define the minimum wavelength, the maximum wavelength and the stepwidth.

Example:

```
int maxElements = 0;
GOMDBTS256_spectralGetSpecmax(handle, &maxElements);
double* values = new double[maxElements];
GOMDBTS256_ spectralGetSpectrumCalibratedWavelength (handle, values);
// do anything you like with the contents of your array e.g.:
cout « "first element =
 « values[0] « endl;
...
cout « "last element = « values[maxElements-1] « endl;
delete [] values;
```

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*spectrum*	Pointer to first element of double array, contains calculated values for spectral unit, size of array is dependent on selected wavelength range and stepwidth.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.2 GOMDBTS256_spectralGetCountsPixel()

```
int GOFUNCDESC GOMDBTS256_spectralGetCountsPixel (
              int handle,
              double * counts )
```

Returns the raw signal values of your spectral unit based on the single pixels of the devices. Memory must be preallocated with size 256 for the array to hold all expected values. Labview users don’t have to care about the array size.

- 1st element contains counts for first pixel
- 2nd element contains counts for second pixel
- ...

Example:

```
double* counts = new double[256];
GOMDBTS256_spectralGetCountsPixel(handle, counts);
// do anything you like with the contents of your array e.g.:
cout « "pixel 0 =
 « counts[0] « endl;
...
cout « "pixel 255 =
 « counts[255] « endl;
delete [] counts;
```

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*counts*	Pointer to first element of double array, contains counts for spectral unit, size of array must be big enough to hold 256 double values.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.3 GOMDBTS256_spectralGetCountsWavelength()

```
int GOFUNCDESC GOMDBTS256_spectralGetCountsWavelength (
              int handle,
              double * counts )
```

Returns the raw signal of your spectral unit. Number of values should be evaluated with “spectralGetSpecmax”, that preallocated array is big enough to hold all expected values. Labview users don’t have to care about the number of values here.
1st element contains counts for first wavelength interpolation point + 0 $*$ stepwidth
2nd element contains counts for first wavelength interpolation point + 1 $*$ stepwidth
...
The wavelength interpolation points are defined by call of “setWavelengthRange”, where you can define the minimum wavelength,
the maximum wavelength and the stepwidth.

Example:

```
int maxElements = 0;
GOMDBTS256_spectralGetSpecmax(handle, &maxElements);
double* counts = new double[maxElements];
GOMDBTS256_spectralGetCountsWavelength(handle, counts);
// do anything you like with the contents of your array e.g.:
cout « "first element = " « counts[0] « endl;
...
cout « "last element = " « counts[maxElements-1] « endl;
```

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*counts*	Pointer to first element of double array, contains counts for spectral unit, size of array is dependent on selected wavelength range and stepwidth

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.4 GOMDBTS256_spectralGetSpecmax()

```
int GOFUNCDESC GOMDBTS256_spectralGetSpecmax (
            int handle,
            int * value )
```

Returns the number of values you get with “spectralGetCountsWavelength” or “spectralGetSpectrumCalibrated↩ Wavelength” The number of values depend on the wavelength range, you can set with “setWavelengthRange” and is calculated with the formula:
[(lastWavelength – firstWavelength) ÷ stepwidth] + 1

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to integer value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.5 GOMDBTS256_spectralGetUnit()

```
int GOFUNCDESC GOMDBTS256_spectralGetUnit (
            int handle,
            int calibrationEntryNumber,
            char * unit )
```

Returns the unit, valid for spectral measurements with a specified calibration entry.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Parameters

in	*calibrationEntryNumber*	integer value, the number of the calibration entry, as defined in your measurement device: • 0 – 9 factory calibrations • 10 – 19 user specific calibrations Not every number has a corresponding valid calibration.
out	*unit*	String value, contains the unit for the specified calibration entry number.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.6 GOMDBTS256_getColor()

```
int GOFUNCDESC GOMDBTS256_getColor (
            int handle,
            double * UpperX,
            double * UpperY,
            double * UpperZ,
            double * x,
            double * y,
            double * us,
            double * vs,
            double * CCT,
            double * domWL )
```

Returns all color informations for last measurement:

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*UpperX*	Pointer to double, value for “ X ”.
out	*UpperY*	Pointer to double, value for “ Y ”.
out	*UpperZ*	Pointer to double, value for “ Z ”.
out	*x*	Pointer to double, value for “ x ”.
out	*y*	Pointer to double, value for “ y ”.
out	*us*	Pointer to double, value for “ u’ ”.
out	*vs*	Pointer to double, value for “ v’ ”.
out	*CCT*	Pointer to double, value for correlated color temperature.
out	*domWL*	Pointer to double, value for dominant wavelength.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.7 GOMDBTS256_getPeak()

```
int GOFUNCDESC GOMDBTS256_getPeak (
            int handle,
            double * lambda,
            double * power )
```

Returns the peak value of the spectrum. It returns the power and the wavelength [nm], where the peak was detected. The unit of the power depends on the selected calibration entry and will be returned by call of “spectralGetUnit”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*lambda*	Pointer to double, wavelength where peak was detected.
out	*power*	Pointer to double, peak value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.8 GOMDBTS256_getFWHM()

```
int GOFUNCDESC GOMDBTS256_getFWHM (
            int handle,
            double * value )
```

Attention

Only for BTS256-E

Returns the FWHM (full width at half maximum) value.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the FWHM value in [nm].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.9 GOMDBTS256_getCenterWavelength()

```
int GOFUNCDESC GOMDBTS256_getCenterWavelength (
              int handle,
              double * value )
```

Attention

Only for BTS256-E

Returns the center wavelength in [nm].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the center wavelength in [nm].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.10 GOMDBTS256_getCentroidWavelength()

```
int GOFUNCDESC GOMDBTS256_getCentroidWavelength (
              int handle,
              double * value )
```

Attention

Only for BTS256-E

Returns the centroid wavelength in [nm].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the centroid wavelength in [nm].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.11 GOMDBTS256_getScotopic()

```
int GOFUNCDESC GOMDBTS256_getScotopic (
             int handle,
             double * value )
```

Attention

Only for BTS256-E

Returns the scotopic value in [lx].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the scotopic value in [lx].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.12 GOMDBTS256_getScotopicPhotopic()

```
int GOFUNCDESC GOMDBTS256_getScotopicPhotopic (
             int handle,
             double * value )
```

Attention

Only for BTS256-E

Returns the ratio between scotopic and photopic value in [lx].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the ratio between scotopic an photopic value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.13 GOMDBTS256_getEVE()

```
int GOFUNCDESC GOMDBTS256_getEVE (
            int handle,
            double * value )
```

Attention

Only for BTS256-E

Returns the EVE value, only valid for measurements with BTS256-E:

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, the EVE value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.14 GOMDBTS256_getFlicker()

```
int GOFUNCDESC GOMDBTS256_getFlicker (
            int handle,
            double * Emax,
            double * Emin,
            double * percent,
            double * index,
            double * frequency )
```

Attention

Only for BTS256-EF

Returns the flicker values, only valid for measurements with BTS256-EF

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*Emax*	Pointer to double value, maxmimum measured intensity during integral measurement.
out	*Emin*	Pointer to double value, minimum measured intensity during integral measurement.
out	*percent*	Pointer to double value, the flicker percentage value 100% * (Emax – Emin)/(Emax + Emin).
out	*index*	Pointer to double value, the flicker index.
out	*frequency*	Pointer to double the flicker frequency.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.15 GOMDBTS256_getFFT()

```
int GOFUNCDESC GOMDBTS256_getFFT (
            int handle,
            double * frequencies,
            double * percentages,
            double * resolution,
            double * frequencyMax )
```

Attention

Only for BTS256-EF

Besides calculation of the flicker parameters, the BTS256-EF also performs an internal FFT (Fast Fourier Transformation). This makes it possible to detect other frequency components in the signal. This method is valid only for measurements with BTS256-EF.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*frequencies*	Pointer to first element of double array, contains up to six different frequency components, which are part of the signal, memory must be allocated to hold 6 elements.
out	*percentages*	Pointer to first element of double array, contains up to six different percentage values corresponding to the frequencies in frequencies array. • percentages[0] belongs to frequencies[0] • percentages[1] belongs to frequencies[1] • ... • percentages[5] belongs to frequencies[5]
out	*resolution*	Pointer to double value.
out	*frequencyMax*	Pointer to double value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.16 GOMDBTS256_getJA10()

```
int GOFUNCDESC GOMDBTS256_getJA10 (
              int handle,
              double * values )
```

Attention

Only for BTS256-EF

This methods collects the JA10 (standard by the California Energy Commission) values from the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*values*	Pointer to double array with size 6, contains the six JA10 values after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.17 GOMDBTS256_getTLA()

```
int GOFUNCDESC GOMDBTS256_getTLA (
              int handle,
              double * PST,
              double * SVM )
```

Attention

Only for BTS256-EF

With the internal FFT (Fast Fourier Transformation) some additional TLA values (Temporal Light Artefacts) are calculated. P_ST and SVM can be collected with this method. Attention the calculation of the PST can be very memory consuming. Make sure enough RAM is available. (e.g calculation of PST for a TLA measurement with 40kHz tact and 180s duration needs about 1GB of RAM)

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*PST*	Pointer to double value, P_ST as defined in CIE TN 006:2016
out	*SVM*	Pointer to double value, SVM as defined in IEC/TR 61547-1:2015

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.18 GOMDBTS256_getAssistFlickerPerception()

```
int GOFUNCDESC GOMDBTS256_getAssistFlickerPerception (
            int handle,
            double * Mp )
```

Attention

Only for BTS256-EF

With the internal FFT (Fast Fourier Transformation) some additional flicker values are calculated. The Flicker Perception Mp (recommended by ASSIST) can be collected with this method

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*Mp*	Pointer to double value, Mp Assist Flicker Perception

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.19 GOMDBTS256_getDeltaUV()

```
int GOFUNCDESC GOMDBTS256_getDeltaUV (
            int handle,
            double * uv )
```

Returns the real delta uv value of your last measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*uv*	Pointer to double value, the limit for calculation of CCT.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.20 GOMDBTS256_getRadiometricValueOverWLRange()

```
int GOFUNCDESC GOMDBTS256_getRadiometricValueOverWLRange (
            int handle,
            double * value )
```

Returns the integral value over the wavelength range for your spectral measurement. The wavelength range must be defined with “setWavelengthRange”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, the radiometric value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.21 GOMDBTS256_getUVValues()

```
int GOFUNCDESC GOMDBTS256_getUVValues (
            int handle,
            double * UV_A,
            double * UV_B,
            double * UV_C )
```

Attention

Only for BTS256-UV-1 and BTS256-UV-2

Returns the values for UV-A, UV-B and UV-C.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*UV_A*	Pointer to double value, the UV-A value.
out	*UV_B*	Pointer to double value, the UV-B value.
out	*UV_C*	Pointer to double value, the UV-C value.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.22 GOMDBTS256_getCRI()

```
int GOFUNCDESC GOMDBTS256_getCRI (
            int handle,
            double * Ra,
            double * R1,
            double * R2,
            double * R3,
            double * R4,
            double * R5,
            double * R6,
            double * R7,
            double * R8,
            double * R9,
            double * R10,
            double * R11,
            double * R12,
            double * R13,
            double * R14 )
```

Color Rendering Index (CRI) (sometimes called color rendition index), is a quantitative measure of the ability of a light source to reproduce the colors of various objects faithfully in comparison with an ideal or natural light source. Light sources with a high CRI are desirable in color-critical applications such as photography and cinematography. It is defined by the International Commission on Illumination as follows:

Color rendering: Effect of an illuminant on the color appearance of objects by conscious or subconscious comparison with their color appearance under a reference illuminant.

This method needs firmware version >= 0.35 but shouldn't be used any more. It's in SDK because of compatibility reasons. Use getCRI2 instead.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*Ra*	Pointer to double, average value of R1 – R8
out	*R1*	Pointer to double, value of R1
out	*R2*	Pointer to double, value of R2
out	*R3*	Pointer to double, value of R3
out	*R4*	Pointer to double, value of R4
out	*R5*	Pointer to double, value of R5
out	*R6*	Pointer to double, value of R6
out	*R7*	Pointer to double, value of R7
out	*R8*	Pointer to double, value of R8
out	*R9*	Pointer to double, value of R9
out	*R10*	Pointer to double, value of R10
out	*R11*	Pointer to double, value of R11
out	*R12*	Pointer to double, value of R12
out	*R13*	Pointer to double, value of R13
out	*R14*	Pointer to double, value of R14

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.23 GOMDBTS256_getCRI2()

```
int GOFUNCDESC GOMDBTS256_getCRI2 (
             int handle,
             double * Ra,
             double * R1,
             double * R2,
             double * R3,
             double * R4,
             double * R5,
             double * R6,
             double * R7,
             double * R8,
             double * R9,
             double * R10,
             double * R11,
             double * R12,
             double * R13,
             double * R14,
             double * R15 )
```

Color Rendering Index (CRI) (sometimes called color rendition index), is a quantitative measure of the ability of a light source to reproduce the colors of various objects faithfully in comparison with an ideal or natural light source. Light sources with a high CRI are desirable in color-critical applications such as photography and cinematography. It is defined by the International Commission on Illumination as follows:
Color rendering: Effect of an illuminant on the color appearance of objects by conscious or subconscious comparison with their color appearance under a reference illuminant.
This method needs firmware version >= 0.48 and should be used instead of “getCRI”. Difference to the old version is the newer specified value R15.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*Ra*	Pointer to double, average value of R1 – R8
out	*R1*	Pointer to double, value of R1
out	*R2*	Pointer to double, value of R2
out	*R3*	Pointer to double, value of R3
out	*R4*	Pointer to double, value of R4
out	*R5*	Pointer to double, value of R5
out	*R6*	Pointer to double, value of R6
out	*R7*	Pointer to double, value of R7
out	*R8*	Pointer to double, value of R8
out	*R9*	Pointer to double, value of R9
out	*R10*	Pointer to double, value of R10
out	*R11*	Pointer to double, value of R11
out	*R12*	Pointer to double, value of R12
out	*R13*	Pointer to double, value of R13
out	*R14*	Pointer to double, value of R14
out	*R15*	Pointer to double, value of R15

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.24 GOMDBTS256_integralProgressionGetNrOfValues()

```
int GOFUNCDESC GOMDBTS256_integralProgressionGetNrOfValues (
            int handle,
            int * value )
```

Attention

Only for BTS256-EF

This method returns the number of values which are stored to show you the chronological sequence of intensity values detected during integral measurement for flicker purpose. This value may change for each single measurement and depends on the integration time for your integral sensor. This method is valid only for measurements with BTS256-EF.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the number of values stored in BTS256-EF internal memory.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.25 GOMDBTS256_integralProgressionGetCalibratedValues()

```
int GOFUNCDESC GOMDBTS256_integralProgressionGetCalibratedValues (
            int handle,
            double * values )
```

Attention

Only for BTS256-EF

This method returns the complete signal progression which shows the chronological sequence of intensity values detected during integral measurement for flicker purpose. This method is valid only for measurements with BT↩ S256-EF.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*values*	Pointer to first element of double array, contains the complete signal progression in chronological order, enough memory must be allocated to hold all stored values, the number of stored values must be evaluated by calling integralProgressionGetNrOfValues. The number of values may change for each measurement.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.26 GOMDBTS256_getBLH()

```
int GOFUNCDESC GOMDBTS256_getBLH (
            int handle,
            double * value )
```

This method returns the Blue Light Hazard value.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, that contains the BLH value after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.27 GOMDBTS256_getMelanopic()

```
int GOFUNCDESC GOMDBTS256_getMelanopic (
            int handle,
            double * Ev,
            double * Eez,
            double * Evmel )
```

This method returns the alpha-opic Melanopic values after CIE S026: Melanopic Irradiance (Ee,mel) and Melanopic Daylight(D65) Equivalent Illuminance (Ev,mel,D65)

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*Ev*	Pointer to double value, that contains the photopic value after return.
out	*Eez*	Pointer to double value, that contains the Melanopic Irradiance after return.
out	*Evmel*	Pointer to double value, that contains the Melanopic Daylight Equivalent Illuminance after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.28 GOMDBTS256_getBilirubin()

```
int GOFUNCDESC GOMDBTS256_getBilirubin (
            int handle,
            double * bilirubinIEC,
            double * bilirubinAAP )
```

This method returns the values for Bilirubin IEC 60601-2-50 and Bilirubin AAP.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*bilirubinIEC*	Pointer to double value, that contains the Bilirubin IEC 60601-2-50 after return.
out	*bilirubinAAP*	Pointer to double value, that contains the Bilirubin AAP after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.7.2.29 GOMDBTS256_getColorCIE170()

```
int GOFUNCDESC GOMDBTS256_getColorCIE170 (
            int handle,
            double * L_MB,
            double * M_MB,
            double * S_MB,
            double * X_F,
            double * Y_F,
            double * Z_F )
```

CIE 170-2 defines a "fundamental chromaticity diagram of which the coordinates correspond to physiologically significant axes" This method returns the coordinates L_MB, M_MB and S_MB of this new chromaricity diagram and the transformations to the tristimulus coordinates X_F, Y_F and Z_F

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*L_MB*	Pointer to a double value, contains L_MB value after return.
out	*M_MB*	Pointer to a double value, contains M_MB value after return.
out	*S_MB*	Pointer to a double value, contains S_MB value after return.
out	*X_F*	Pointer to a double value, contains X_F value after return.
out	*Y_F*	Pointer to a double value, contains Y_F value after return.
out	*Z_F*	Pointer to a double value, contains Z_F value after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8 Methods to control internal logger

Functions

- int GOFUNCDESC GOMDBTS256_loggerReadData (int handle, int index, int ∗x, int ∗y, char ∗text)
- int GOFUNCDESC GOMDBTS256_loggerDeleteData (int handle, int index)
- int GOFUNCDESC GOMDBTS256_loggerArrayDiodeSetIndex (int handle, int startIndex, int stopIndex)
- int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetIndex (int handle, int ∗startIndex, int ∗stopIndex)
- int GOFUNCDESC GOMDBTS256_loggerArrayDiodeSetTiming (int handle, int timingInSeconds)
- int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetTiming (int handle, int ∗timingInSeconds)
- int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetLastValidIndex (int handle, int ∗index)
- int GOFUNCDESC GOMDBTS256_loggerDLIReadData (int handle, int index, int ∗diodeIdxStart, int ∗diode↩ IdxStop, char ∗startTime, double ∗tactRate, double ∗doseValue, char ∗doseUnit)
- int GOFUNCDESC GOMDBTS256_loggerDiodeSetIndex (int handle, int startIndex, int stopIndex)
- int GOFUNCDESC GOMDBTS256_loggerDiodeGetIndex (int handle, int ∗startIndex, int ∗stopIndex)
- int GOFUNCDESC GOMDBTS256_loggerDiodeSetTiming (int handle, int timingIn100ms)
- int GOFUNCDESC GOMDBTS256_loggerDiodeGetTiming (int handle, int ∗timingIn100ms)
- int GOFUNCDESC GOMDBTS256_loggerDiodeGetLastValidIndex (int handle, int ∗index)
- int GOFUNCDESC GOMDBTS256_loggerDiodeReadData (int handle, int indexFrom, int indexTo, double ∗values)
- int GOFUNCDESC GOMDBTS256_loggerDiodeReadData2 (int handle, int indexFrom, int indexTo, double ∗values, int ∗indexArray)
- int GOFUNCDESC GOMDBTS256_loggerDiodeDeleteData (int handle, int index)
- int GOFUNCDESC GOMDBTS256_loggerGetTemperature (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_loggerGetRTCTime (int handle, int ∗day, int ∗month, int ∗year, int ∗hour, int ∗minute, int ∗second)
- int GOFUNCDESC GOMDBTS256_loggerAreaSetIndex (int handle, int startIndex, int stopIndex)
- int GOFUNCDESC GOMDBTS256_loggerAreaGetIndex (int handle, int ∗startIndex, int ∗stopIndex)

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 GOMDBTS256_loggerReadData()

```
int GOFUNCDESC GOMDBTS256_loggerReadData (
            int handle,
            int index,
            int * x,
            int * y,
            char * text )
```

Attention

Only for BTS256-E

Read all logger data for a specific index from the device. data can then be collect the same way as after a measurement.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*index*	integer value, the index of the logger
out	*x*	pointer to integer, the x value of this logger index
out	*y*	pointer to integer, the y value of this logger index
out	*text*	string value; contains the text for this logger index

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.2 GOMDBTS256_loggerDeleteData()

```
int GOFUNCDESC GOMDBTS256_loggerDeleteData (
            int handle,
            int index )
```

Attention

Only for BTS256-E

Deletes the logger data for a specific index from the device

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*index*	integer value, the index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.3 GOMDBTS256_loggerArrayDiodeSetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerArrayDiodeSetIndex (
            int handle,
            int startIndex,
            int stopIndex )
```

Attention

Only for BTS256-E

Sets the start and stop index for the Array+Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*startIndex*	integer value, the start index of the logger
in	*stopIndex*	integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.4 GOMDBTS256_loggerArrayDiodeGetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetIndex (
             int handle,
             int * startIndex,
             int * stopIndex )
```

Attention

Only for BTS256-E

Returns the actual start and stop index of the Array+Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*startIndex*	pointer to integer value, the start index of the logger
out	*stopIndex*	pointer to integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.5 GOMDBTS256_loggerArrayDiodeSetTiming()

```
int GOFUNCDESC GOMDBTS256_loggerArrayDiodeSetTiming (
            int handle,
            int timingInSeconds )
```

Attention

Only for BTS256-E

Sets the timing for the Array+Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*timingInSeconds*	integer value, timing in seconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.6 GOMDBTS256_loggerArrayDiodeGetTiming()

```
int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetTiming (
            int handle,
            int * timingInSeconds )
```

Attention

Only for BTS256-E

Returns the timing for the Array+Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timingInSeconds*	pointer to integer value, timing in seconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.7 GOMDBTS256_loggerArrayDiodeGetLastValidIndex()

```
int GOFUNCDESC GOMDBTS256_loggerArrayDiodeGetLastValidIndex (
             int handle,
             int * index )
```

Attention

Only for BTS256-E

Returns the last valid index of the Array+Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*index*	pointer to integer value, last valid index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.8 GOMDBTS256_loggerDLIReadData()

```
int GOFUNCDESC GOMDBTS256_loggerDLIReadData (
             int handle,
             int index,
             int * diodeIdxStart,
             int * diodeIdxStop,
             char * startTime,
             double * tactRate,
             double * doseValue,
             char * doseUnit )
```

Attention

Only for BTS256-UV

A DLI measurement conatins of serveral integral and spectral measurement in a certain interval with a specified tactRate. This methods reads the DLI logger data for a specific index from the device. It returns the dose value calculated from all related integral measurements and the start and stop index of the these measurements. The single integral measurements, can then be collected with the method GOMDBTS256_loggerDiodeReadData().

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*index*	integer value, the index of the DLI logger
out	*diodeIdxStart*	pointer to integer, the start index of the diode logger
out	*diodeIdxStop*	pointer to integer, the stop index of the diode logger
out	*startTime*	string value, contains the start time of the logger, must be allocated with 12 bytes
out	*tactRate*	double value, contains the measurement tact in seconds
out	*doseValue*	double value, contains the dose value of the logger mesurement
out	*doseUnit*	string value, contains the dose unit of the logger mesurement, must be allocated with 8 bytes

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.9 GOMDBTS256_loggerDiodeSetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeSetIndex (
            int handle,
            int startIndex,
            int stopIndex )
```

Attention

Only for BTS256-E

Sets the start and stop index for the Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*startIndex*	integer value, the start index of the logger
in	*stopIndex*	integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.10 GOMDBTS256_loggerDiodeGetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeGetIndex (
            int handle,
            int * startIndex,
            int * stopIndex )
```

Attention

Only for BTS256-E

Returns the actual start and stop index of the Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*startIndex*	pointer to integer value, the start index of the logger
out	*stopIndex*	pointer to integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.11 GOMDBTS256_loggerDiodeSetTiming()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeSetTiming (
            int handle,
            int timingIn100ms )
```

Attention

Only for BTS256-E

Sets the timing for the Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*timingIn100ms*	integer value, timing in 100 miliseconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.12 GOMDBTS256_loggerDiodeGetTiming()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeGetTiming (
            int handle,
            int * timingIn100ms )
```

Attention

Only for BTS256-E

Returns the timing for the Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*timingIn100ms*	pointer to integer value, timing in 100 miliseconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.13 GOMDBTS256_loggerDiodeGetLastValidIndex()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeGetLastValidIndex (
            int handle,
            int * index )
```

Attention

Only for BTS256-E

Returns the last valid index of the Diode Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*index*	pointer to integer value, last valid index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.14 GOMDBTS256_loggerDiodeReadData()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeReadData (
            int handle,
            int indexFrom,
            int indexTo,
            double * values )
```

Attention

Only for BTS256-E

Reads out the fast logger memory and returns all measurements of the integral device

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*indexFrom*	integer value, the start index of the logger
in	*indexTo*	integer value, the stop index of the logger
out	*values*	double array size of (indexTo-indexFrom+1). Contains the doide values of the logger for specified indicies.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.15 GOMDBTS256_loggerDiodeReadData2()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeReadData2 (
            int handle,
            int indexFrom,
            int indexTo,
            double * values,
            int * indexArray )
```

Attention

Only for BTS256-E

Reads out the fast logger memory and returns all measurements of the integral device. In addition to GOMDBTS256_loggerDiodeReadData() it also returns the information if a corresponding array measurement is available an its index.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*indexFrom*	integer value, the start index of the logger
in	*indexTo*	integer value, the stop index of the logger
out	*values*	double array size of (indexTo-indexFrom+1). Contains the doide values of the logger for specified indicies.
out	*indexArray*	int array size of (indexTo-indexFrom+1). Contains the index of the corresponding array measurement in the array logger (-1 if not available).

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.16 GOMDBTS256_loggerDiodeDeleteData()

```
int GOFUNCDESC GOMDBTS256_loggerDiodeDeleteData (
            int handle,
            int index )
```

Attention

Only for BTS256-E

Deletes the index oft the fast logger memory and marks it as invalid.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*index*	Integer value; the index to be deleted.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.17 GOMDBTS256_loggerGetTemperature()

```
int GOFUNCDESC GOMDBTS256_loggerGetTemperature (
            int handle,
            double * value )
```

Attention

Only for BTS256-E

This method returns the temperature of actual loaded logger index.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the temperature in [℃].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.18 GOMDBTS256_loggerGetRTCTime()

```
int GOFUNCDESC GOMDBTS256_loggerGetRTCTime (
            int handle,
            int * day,
            int * month,
            int * year,
            int * hour,
            int * minute,
            int * second )
```

Attention

Only for BTS256-E

This method returns the runtime clock value of the actual loaded logger index.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*day*	Pointer to int value, contains the day (1-31).
out	*month*	Pointer to int value, contains the month (1-12).
out	*year*	Pointer to int value, contains the year.
out	*hour*	Pointer to int value, contains the hour (0-23).
out	*minute*	Pointer to int value, contains the minute (0-59).
out	*second*	Pointer to int value, contains the second (0-59).

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.19 GOMDBTS256_loggerAreaSetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerAreaSetIndex (
            int handle,
            int startIndex,
            int stopIndex )
```

Attention

Only for BTS256-E

Sets the start and stop index for the Area Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*startIndex*	integer value, the start index of the logger
in	*stopIndex*	integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.8.2.20 GOMDBTS256_loggerAreaGetIndex()

```
int GOFUNCDESC GOMDBTS256_loggerAreaGetIndex (
            int handle,
            int * startIndex,
            int * stopIndex )
```

Attention

Only for BTS256-E

Returns the actual start and stop index of the Area Logger

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*startIndex*	pointer to integer value, the start index of the logger
out	*stopIndex*	pointer to integer value, the stop index of the logger

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9 Methods for area calculation

Functions

- int GOFUNCDESC GOMDBTS256_areaSetWall (int handle, int wallNumber, int x1, int y1, int x2, int y2)
- int GOFUNCDESC GOMDBTS256_areaGetWall (int handle, int wallNumber, int ∗x1, int ∗y1, int ∗x2, int ∗y2)
- int GOFUNCDESC GOMDBTS256_areaDeleteWall (int handle, int wallNumber)
- int GOFUNCDESC GOMDBTS256_areaSetMeasurePoint (int handle, int pointNumber, int x1, int y1, int areaNumber, char ∗text)
- int GOFUNCDESC GOMDBTS256_areaGetMeasurePoint (int handle, int pointNumber, int ∗x1, int ∗y1, int ∗areaNumber, char ∗text)
- int GOFUNCDESC GOMDBTS256_areaDeleteMeasurePoint (int handle, int pointNumber)
- int GOFUNCDESC GOMDBTS256_areaSave (int handle, int areaNumber, char ∗text)
- int GOFUNCDESC GOMDBTS256_areaLoad (int handle, int areaNumber, char ∗text)
- int GOFUNCDESC GOMDBTS256_areaDelete (int handle, int areaNumber)

5.9.1 Detailed Description

Attention

These group of methods are only available for BTS256-E device

5.9.2 Function Documentation

5.9.2.1 GOMDBTS256_areaSetWall()

```
int GOFUNCDESC GOMDBTS256_areaSetWall (
            int handle,
            int wallNumber,
            int x1,
            int y1,
            int x2,
            int y2 )
```

Attention

Only for BTS256-E

Sets a wall for display within area logger mode; you can define up to 20 walls; the walls are defined by a rectangle with a top-left start point and a bottom-right stop point within a display of 228 x 130 pixel.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*wallNumber*	Integer value; range: 0 - 19
in	*x1*	Integer value; start coordinate x1, range: 0 – 228, x1 must be $<$ x2
in	*y1*	Integer value; start coordinate y1, range: 0 – 130, y1 must be $<$ y2
in	*x2*	Integer value; stop coordinate x2, range: 0 – 228, x2 must be $>$ x1
in	*y2*	Integer value; stop coordinate y2, range: 0 – 130, y2 must be $>$ y2

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.2 GOMDBTS256_areaGetWall()

```
int GOFUNCDESC GOMDBTS256_areaGetWall (
            int handle,
            int wallNumber,
            int * x1,
            int * y1,
            int * x2,
            int * y2 )
```

Attention

Only for BTS256-E

Returns the coordinates of a wall for display within area logger mode; you can select one of 20 walls; the wall definition is the top-left start point and the bottom-right stop point of the surrounding rectangle.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*wallNumber*	Integer value; range: 0 - 19
out	*x1*	pointer to integer value; start coordinate x1, range: 0 – 228
out	*y1*	pointer to integer value; start coordinate y1, range: 0 – 130
out	*x2*	pointer to integer value; stop coordinate x2, range: 0 – 228
out	*y2*	pointer to integer value; stop coordinate y2, range: 0 – 130

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.3 GOMDBTS256_areaDeleteWall()

```
int GOFUNCDESC GOMDBTS256_areaDeleteWall (
            int handle,
            int wallNumber )
```

Attention

Only for BTS256-E

Deletes a previous defined wall. You can select one of 20 walls.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*wallNumber*	Integer value; range: 0 - 19

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.4 GOMDBTS256_areaSetMeasurePoint()

```
int GOFUNCDESC GOMDBTS256_areaSetMeasurePoint (
            int handle,
            int pointNumber,
            int x1,
            int y1,
            int areaNumber,
            char * text )
```

Attention

Only for BTS256-E

sets a wall for display within area logger mode; you can define up to 20 walls; the walls are defined by a rectangle with a top-left start point and a bottom-right stop point

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pointNumber*	Integer value; range: 0 - 99
in	*x1*	Integer value; start coordinate x1, range: 0 – 228
in	*y1*	integer value; start coordinate y1, range: 0 – 130
in	*areaNumber*	Integer value; always has to be set to “0”
in	*text*	String value; maximum 16 characters; name of the measurement point

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.5 GOMDBTS256_areaGetMeasurePoint()

```
int GOFUNCDESC GOMDBTS256_areaGetMeasurePoint (
            int handle,
```

```
            int pointNumber,
            int * x1,
            int * y1,
            int * areaNumber,
            char * text )
```

Attention

Only for BTS256-E

Returns the coordinates of a wall for display within area logger mode; you can select one of 20 walls; the wall definition is the top-left start point and the bottom-right stop point of the surrounding rectangle

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pointNumber*	Integer value; range: 0 - 99
out	*x1*	Pointer to integer value; start coordinate x1, range: 0 – 228
out	*y1*	Pointer to integer value; start coordinate y1, range: 0 – 130
out	*areaNumber*	Pointer to integer value; always “0”
out	*text*	String value; contains the defined name for the measurement point

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.6 GOMDBTS256_areaDeleteMeasurePoint()

```
int GOFUNCDESC GOMDBTS256_areaDeleteMeasurePoint (
            int handle,
            int pointNumber )
```

Attention

Only for BTS256-E

Deletes a previous defined point. You can select one of 100 points.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pointNumber*	Integer value; range: 0 - 99

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.7 GOMDBTS256_areaSave()

```
int GOFUNCDESC GOMDBTS256_areaSave (
            int handle,
            int areaNumber,
            char * text )
```

Attention

Only for BTS256-E

This method saves the area.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*areaNumber*	Integer value; has to be set to “0”
in	*text*	String value; the name, you want to define for the area

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.8 GOMDBTS256_areaLoad()

```
int GOFUNCDESC GOMDBTS256_areaLoad (
            int handle,
            int areaNumber,
            char * text )
```

Attention

Only for BTS256-E

This method loads the area.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*areaNumber*	Integer value; has to be set to “0”
out	*text*	string value; the name, you defined for the area

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.9.2.9 GOMDBTS256_areaDelete()

```
int GOFUNCDESC GOMDBTS256_areaDelete (
            int handle,
            int areaNumber )
```

Attention

Only for BTS256-E

This method delets the area.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*areaNumber*	Integer value; has to be set to “0”

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.10 User weightings methods

5.11 Substitution methods

Functions

- int GOFUNCDESC GOMDBTS256_substitutionEnableCorrection (int handle, bool active)
- int GOFUNCDESC GOMDBTS256_substitutionIsEnabledCorrection (int handle, bool ∗active)
- int GOFUNCDESC GOMDBTS256_substitutionLoadFactors (int handle, char ∗absoluteFileName)
- int GOFUNCDESC GOMDBTS256_substitutionSaveFactors (int handle, char ∗absoluteFileName)
- int GOFUNCDESC GOMDBTS256_substitutionGetLoadedFilename (int handle, char ∗absoluteFileName)
- int GOFUNCDESC GOMDBTS256_substitutionMeasurementWithoutTestDevice (int handle, bool is↩ ExternalSphere, double ∗saturation, double ∗counts, double ∗current)
- int GOFUNCDESC GOMDBTS256_substitutionMeasurementWithTestDevice (int handle, bool isExternal↩ Sphere, double ∗saturation, double ∗counts, double ∗current)
- int GOFUNCDESC GOMDBTS256_substitutionGetIntegralFactor (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetPresetIntegralFactor (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetSpectralFactor (int handle, int pixelNumber, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetSpectralFactors (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetPresetSpectralFactor (int handle, int pixelNumber, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetPresetSpectralFactors (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGetDateTime (int handle, int ∗day, int ∗month, int ∗year, int ∗hh, int ∗mm, int ∗ss)
- int GOFUNCDESC GOMDBTS256_substitutionGetDateTime2 (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_substitutionSetComment (int handle, char ∗comment)
- int GOFUNCDESC GOMDBTS256_substitutionGetComment (int handle, char ∗comment)
- int GOFUNCDESC GOMDBTS256_substitutionSetIntegrationTimeInMs (int handle, int value)
- int GOFUNCDESC GOMDBTS256_substitutionGetIntegrationTimeInMs (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_substitutionSetDynamicTimeMode (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_substitutionGetDynamicTimeMode (int handle, bool ∗value)

5.11.1 Detailed Description

methods for self absorbtion correction (substitution) with a device under test.

Following methods handle self absorbtion correction. This is necessary when using integrating spheres. These spheres are calibrated when empty. By including a test object and its holder, the properties of the sphere are changed and measuring results are falsified. Therefore, for different test objects, the self-absorption correction must first be performed, before performing a measurement. Factor determination, however, only has to be done once for the corresponding test objects as the factors can be saved and reactivated.
A second falsification of measurement results results from changes to the sphere geometry. For changes to the sphere geometry, a self-absorption correction must also be performed. There are two groups of methods for the two self-absorption types. (Self-absorption correction DUT and self-absorption correction GEO (geometry)).

5.11.2 C++ Calling Example

measure self absorbtion correction with DUT and activate it.

```
int handle;
double saturation, current, counts[256];
GOMDBTS256_getHandle(NULL, &handle);
       //initialization
GOMDBTS256_substitutionMeasurementWithTestDevice(handle, true, &saturation, counts, &current);
       //measurment with DUT in Sphere(lamp on!)
GOMDBTS256_substitutionMeasurementWithoutTestDevice(handle, true, &saturation, counts, &current);
       //measurment without DUT (lamp on!)
GOMDBTS256_substitutionEnableCorrection(handle, true);                                          //enable
GOMDBTS256_releaseHandle(handle);
       //release handle
```

5.11.3 Function Documentation

5.11.3.1 GOMDBTS256_substitutionEnableCorrection()

```
int GOFUNCDESC GOMDBTS256_substitutionEnableCorrection (
            int handle,
            bool active )
```

This method enables and disables substitution correction of the device under test (DUT). Substitution correction is an important step when performing measurements with an integrating sphere. The calibration values which are saved in the BTS2048 for measurement of the luminous flux are determined using an empty sphere. As soon as the geometriy of the sphere is changed, the reflection properties of the sphere change. This change, which would result in a measurement error, must be compensated for. This is done through substitution correction.
For this method to be effective, substitution factors for the measurement object must be determined before the measurement. This is done using the “substitutionMeasurementWithoutDevice” and “substitutionMeasurement↩ WithDevice” methods. These factors can be saved (“substitutionLoadFactors”) and later loaded (“substitution↩ LoadFactors”). Initially, all factors are saved with “1.0”. This is the neutral factor that does not lead to correction.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*active*	Boolean value that determines if the substitution correction will be activated or not: • true: Substitution correction activated • false: Substitution correction deactivated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.2 GOMDBTS256_substitutionIsEnabledCorrection()

```
int GOFUNCDESC GOMDBTS256_substitutionIsEnabledCorrection (
            int handle,
            bool * active )
```

Checks if substitution is activated for use in measurements.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*active*	Pointer to boolean value: • true if substitution correction is active • false if not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.3 GOMDBTS256_substitutionLoadFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionLoadFactors (
            int handle,
            char * absoluteFileName )
```

Loads the previous measured and saved substitution factors from file into measurement device. These factors will be used when substitution correction is set enabled by call of “substitutionEnableCorrection”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*absoluteFileName*	String value, complete path to file.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.4 GOMDBTS256_substitutionSaveFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionSaveFactors (
            int handle,
            char * absoluteFileName )
```

Actual measured substitution correction factors will be stored in specified file. File name has to be complete path to file.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*absoluteFileName*	String value, complete path to target file.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.5 GOMDBTS256_substitutionGetLoadedFilename()

```
int GOFUNCDESC GOMDBTS256_substitutionGetLoadedFilename (
            int handle,
            char * absoluteFileName )
```

Returns the name of the file you loaded substitution correction factors from.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*absoluteFileName*	String value, memory must be allocated with 2048 bytes to hold complete file name.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.6 GOMDBTS256_substitutionMeasurementWithoutTestDevice()

```
int GOFUNCDESC GOMDBTS256_substitutionMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This is always the first measurement which have to be done, when new substitution correction values should be created. The empty sphere always has to be measured with this method before calling the next method “substitution↩ MeasurementWithTestDevice“. You must specify, if performing the measurement for use with an internal or external sphere, because internal auxiliary LED will be switched on/off automatically, when an internal sphere is used. For substitution correction measurement for use with an external sphere you have to switch on the auxiliary lamp, delivered with your sphere, manually.
Always use same integration time for measurement of empty and filled sphere.
Always try to set integration time in that way, that saturation results between 54% and 95% for both measurements. Otherwise evaluated correction factors may be not so good.
Remove all samples from the sphere (internal or external sphere) and call method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value: • true is using external sphere • false is not using external sphere
out	*saturation*	Pointer to double value, contains the saturation for spectral device in [%].
out	*counts*	Array of double values, size = 256, contains the raw counts for spectral device.
out	*current*	Pointer to double value, contains the raw value for integral device.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.7 GOMDBTS256_substitutionMeasurementWithTestDevice()

```
int GOFUNCDESC GOMDBTS256_substitutionMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should be called second to compute substitution factors of the DUT. Switch on your auxiliary lamp and wait for the burn in time predefined by the manufacturer in order to obtain a stable signal.
You must specify, if performing the measurement for use with an internal or external sphere, because internal auxiliary LED will be switched on/off automatically, when an internal sphere is used. For substitution correction measurement for use with an external sphere you have to switch on the auxiliary lamp, delivered with your sphere, manually.
Always use same integration time for measurement of empty and filled sphere.
Always try to set integration time in that way, that saturation results between 54% and 95% for both measurements. Otherwise evaluated correction factors may be not so good.
Place the geometry sample into the sphere (internal or external sphere) and call method.
“substitutionMeasurementWithoutTestDevice” must be done every time before calling this method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value: • true is using external sphere • false is not using external sphere
out	*saturation*	Pointer to double value, contains the saturation for spectral device in [%].
out	*counts*	Array of double values, size = 256, contains the raw counts for spectral device.
out	*current*	Pointer to double value, contains the raw value for integral device.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.8 GOMDBTS256_substitutionGetIntegralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGetIntegralFactor (
            int handle,
            double * factor )
```

Returns the calibration correction factor for the integral device.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to double, contains the correction factor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.9 GOMDBTS256_substitutionGetPresetIntegralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGetPresetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the current substitution factor for the integral sensor. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGetPresetIntegralFactor method should be used.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to a double value, contains the substitution factor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.10 GOMDBTS256_substitutionGetSpectralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

This method provides the current substitution factor for a specific pixel of the spectrometer. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGetPresetSpectralFactor method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 255.
out	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.11 GOMDBTS256_substitutionGetSpectralFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all 256 current substitution factors of the spectrometer. These factors are used in the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factors that are currently present (regardless of whether substitution correction is active or not) are determined, the substitutionGetPresetSpectralFactors method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to the first Value of a Double Array, containing the substitution factors of all pixel. Allocated with 256 values

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.12 GOMDBTS256_substitutionGetPresetSpectralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGetPresetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

This method provides the default substitution factor for a specific pixel of the spectrometer, regardless of whether the substitution correction is on or not. This factor only applys when the substitution correction is activated. The substitutionGetSpectralFactor method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 2047.
out	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.13 GOMDBTS256_substitutionGetPresetSpectralFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGetPresetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all the default substitution factors for the spectral sensor, regardless of whether the substitution correction is on or not. These factors only apply when the substitution correction is activated. The substitutionGet↩ SpectralFactor method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to the first value of a Double array, containing the substitution factors of all pixel. Allocated with 256 values

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.14 GOMDBTS256_substitutionGetDateTime()

```
int GOFUNCDESC GOMDBTS256_substitutionGetDateTime (
            int handle,
            int * day,
            int * month,
            int * year,
            int * hh,
            int * mm,
            int * ss )
```

This method returns the date and time the current substitution corrction was stored.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*day*	Pointer to integer, returns the day of the date
out	*month*	Pointer to integer, returns the month of the date
out	*year*	Pointer to Integer, contains the year of the date
out	*hh*	Pointer to integer, contains the hours of the time after the jump
out	*mm*	Pointer to integer, contains the minutes of the time after the jump
out	*ss*	Pointer to integer, returns the seconds of the time

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.15 GOMDBTS256_substitutionGetDateTime2()

```
int GOFUNCDESC GOMDBTS256_substitutionGetDateTime2 (
            int handle,
            double * value )
```

This method returns the date and time the current substitution corrction was stored as timestamp.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double, contains the time as timestamp

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.16 GOMDBTS256_substitutionSetComment()

```
int GOFUNCDESC GOMDBTS256_substitutionSetComment (
            int handle,
            char * comment )
```

This method sets a comment describing the current substitution more closely. It should be called before the current substitution correction is saved, since this comment is also stored.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*comment*	null-terminated string containing the comment; Maximum permissible length including terminator: 1024 bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.17 GOMDBTS256_substitutionGetComment()

```
int GOFUNCDESC GOMDBTS256_substitutionGetComment (
            int handle,
            char * comment )
```

This method returns the comment set by the substitutionSetComment method.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*comment*	null-terminated string, must be allocated with 1024 bytes; contains the comment

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.18 GOMDBTS256_substitutionSetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_substitutionSetIntegrationTimeInMs (
            int handle,
            int value )
```

With this method it is possible to set the integration time in µs, which is used for the substitution measurement. If the integration time is selected to high, the value can be overloaded and not usable. For automatic determination of the integration time GOMDBTS256_substitutionSetDynamicTimeMode() can be used.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Integer value, integration time in miliseconds [ms].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.19 GOMDBTS256_substitutionGetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_substitutionGetIntegrationTimeInMs (
            int handle,
            int * value )
```

Returns the last selcted integration time in ms.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to integer value, contains the integration time in miliseconds [ms].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.20 GOMDBTS256_substitutionSetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_substitutionSetDynamicTimeMode (
            int handle,
            bool value )
```

This method activates the dynamic adjustment of the integration time for the substitution.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean Value: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.11.3.21 GOMDBTS256_substitutionGetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_substitutionGetDynamicTimeMode (
            int handle,
            bool * value )
```

This method returns the dynamic time mode for the substitution.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean; containing the „time mode“: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12 Substitution geometry methods

Functions

- int GOFUNCDESC GOMDBTS256_substitutionGeoEnableCorrection (int handle, bool active)
- int GOFUNCDESC GOMDBTS256_substitutionGeoIsEnabledCorrection (int handle, bool ∗active)
- int GOFUNCDESC GOMDBTS256_substitutionGeoLoadFactors (int handle, char ∗absoluteFileName)
- int GOFUNCDESC GOMDBTS256_substitutionGeoSaveFactors (int handle, char ∗absoluteFileName)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetLoadedFilename (int handle, char ∗absoluteFile↩ Name)
- int GOFUNCDESC GOMDBTS256_substitutionGeoMeasurementWithoutTestDevice (int handle, bool is↩ ExternalSphere, double ∗saturation, double ∗counts, double ∗current)
- int GOFUNCDESC GOMDBTS256_substitutionGeoMeasurementWithTestDevice (int handle, bool is↩ ExternalSphere, double ∗saturation, double ∗counts, double ∗current)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetIntegralFactor (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetIntegralFactor (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetSpectralFactor (int handle, int pixelNumber, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetSpectralFactors (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetSpectralFactor (int handle, int pixelNumber, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetSpectralFactors (int handle, double ∗factor)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetDateTime (int handle, int ∗day, int ∗month, int ∗year, int ∗hh, int ∗mm, int ∗ss)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetDateTime2 (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_substitutionGeoSetComment (int handle, char ∗comment)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetComment (int handle, char ∗comment)
- int GOFUNCDESC GOMDBTS256_substitutionGeoSetIntegrationTimeInMs (int handle, int value)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetIntegrationTimeInMs (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_substitutionGeoSetDynamicTimeMode (int handle, bool value)
- int GOFUNCDESC GOMDBTS256_substitutionGeoGetDynamicTimeMode (int handle, bool ∗value)

5.12.1 Detailed Description

methods for self absorbtion correction (substitution) with the sphere geometrie

The self absorbtion correction GEO handles the correction for changes to the sphere geometry. For a detailed description read the section self absorbtion correction DUT.

5.12.2 Function Documentation

5.12.2.1 GOMDBTS256_substitutionGeoEnableCorrection()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoEnableCorrection (
            int handle,
            bool active )
```

This method enables and disables substitution correction of the geometrie. Substitution correction is an important step when performing measurements with an integrating sphere. The calibration values which are saved in the BTS2048 for measurement of the luminous flux are determined using an empty sphere. As soon as the geometriy of the sphere is changed, the reflection properties of the sphere change. This change, which would result in a measurement error, must be compensated for. This is done through substitution correction.
For this method to be effective, substitution factors for the measurement object must be determined before the measurement. This is done using the “substitutionGeoMeasurementWithoutDevice” and “substitutionGeoMeasurementWithDevice” methods. These factors can be saved (“substitutionGeoLoadFactors”) and later loaded (“substitutionGeoLoadFactors”). Initially, all factors are saved with “1.0”. This is the neutral factor that does not lead to correction.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*active*	Boolean value that determines if the substitution correction will be activated or not: • true: Substitution correction activated • false: Substitution correction deactivated

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.2 GOMDBTS256_substitutionGeoIsEnabledCorrection()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoIsEnabledCorrection (
            int handle,
            bool * active )
```

Checks if substitution correction of the geometrie is enabled or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*active*	Pointer to a boolean value, contains information on whether the substitution correction is active or not: • true: Substitution correction is active • false: Substitution correction is not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.3 GOMDBTS256_substitutionGeoLoadFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoLoadFactors (
            int handle,
            char * absoluteFileName )
```

Loads the previously measured and saved substitution factors from file into measurement device. These factors will be used when substitution correction is enabled by calling the “substitutionEnableCorrection” method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*absoluteFileName*	Zero terminated string, contains the complete path to the file to be loaded.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.4 GOMDBTS256_substitutionGeoSaveFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoSaveFactors (
            int handle,
            char * absoluteFileName )
```

Currently computed substitution factors are saved under the specified file name. The file name must be specified as a complete path.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*absoluteFileName*	String value, complete path to target file.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.5 GOMDBTS256_substitutionGeoGetLoadedFilename()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetLoadedFilename (
            int handle,
            char * absoluteFileName )
```

Returns the file name with the complete file path from which the current substitution factors should be loaded. If the substitution factors are not loaded from the file, an empty string is returned.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*absoluteFileName*	Zero terminated string, must be allocated 2048 bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.6 GOMDBTS256_substitutionGeoMeasurementWithoutTestDevice()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This is always the first measurement which have to be done, when new substitution correction values of the geometrie should be created. The empty sphere always has to be measured with this method before calling the next method “substitutionGeoMeasurementWithTestDevice“. You must specify, if performing the measurement for use with an internal or external sphere, because internal auxiliary LED will be switched on/off automatically, when an internal sphere is used. For substitution correction measurement for use with an external sphere you have to switch on the auxiliary lamp, delivered with your sphere, manually.
Always use same integration time for measurement of empty and filled sphere.
Always try to set integration time in that way, that saturation results between 54% and 95% for both measurements. Otherwise evaluated correction factors may be not so good.
Remove all samples from the sphere (internal or external sphere) and call method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value: • true is using external sphere • false is not using external sphere
out	*saturation*	Pointer to double value, contains the saturation for spectral device in [%].
out	*counts*	Array of double values, size = 256, contains the raw counts for spectral device.
out	*current*	Pointer to double value, contains the raw value for integral device.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.7 GOMDBTS256_substitutionGeoMeasurementWithTestDevice()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should be called second to compute substitution factors of the geometrie. Switch on your auxiliary lamp and wait for the burn in time predefined by the manufacturer in order to obtain a stable signal.
You must specify, if performing the measurement for use with an internal or external sphere, because internal auxiliary LED will be switched on/off automatically, when an internal sphere is used. For substitution correction measurement for use with an external sphere you have to switch on the auxiliary lamp, delivered with your sphere, manually.
Always use same integration time for measurement of empty and filled sphere.
Always try to set integration time in that way, that saturation results between 54% and 95% for both measurements. Otherwise evaluated correction factors may be not so good.
Place the geometry sample into the sphere (internal or external sphere) and call method.
“substitutionGeoMeasurementWithoutTestDevice” must be done every time before calling this method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value: • true is using external sphere • false is not using external sphere
out	*saturation*	Pointer to double value, contains the saturation for spectral device in [%].
out	*counts*	Array of double values, size = 256, contains the raw counts for spectral device.
out	*current*	Pointer to double value, contains the raw value for integral device.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.8 GOMDBTS256_substitutionGeoGetIntegralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the current substitution factor for the integral sensor. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGetPresetIntegralFactor method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to a double value, contains the substitution factor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.9 GOMDBTS256_substitutionGeoGetPresetIntegralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the default substitution factor for the integral sensor, regardless of whether the substitution correction is on or not. This factor is only used when the substitution correction is activated. The method substitutionGetIntegralFactor should be used for the correction factor actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to Double, containing hte substitutionfactor.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.10 GOMDBTS256_substitutionGeoGetSpectralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

This method provides the current substitution factor for a specific pixel of the spectrometer. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGeoGetPresetSpectralFactor method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 255.
out	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.11 GOMDBTS256_substitutionGeoGetSpectralFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all 256 current substitution factors for a specific pixel of the spectrometer. These factors are used in the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factors that are currently present (regardless of whether substitution correction is active or not) are determined, the substitutionGeoGetPresetSpectralFactors method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to the first Value of a Double Array, containing the substitution factors of all pixel. Allocated with 256 values

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.12 GOMDBTS256_substitutionGeoGetPresetSpectralFactor()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

This method provides the default substitution factor for a specific pixel of the spectrometer, regardless of whether the substitution correction is on or not. This factor only applys when the substitution correction is activated. The substitutionGeoGetSpectralFactor method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 255.
out	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.13 GOMDBTS256_substitutionGeoGetPresetSpectralFactors()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetPresetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all the default substitution factors for the spectral sensor, regardless of whether the substitution correction is on or not. These factors only apply when the substitution correction is activated. The substitutionGeo↩ GetSpectralFactors method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*factor*	Pointer to the first value of a Double array, containing the substitution factors of all pixel. Allocated with 256 values

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.14 GOMDBTS256_substitutionGeoGetDateTime()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetDateTime (
            int handle,
            int * day,
            int * month,
            int * year,
            int * hh,
            int * mm,
            int * ss )
```

This method returns the date and timethe current substitution corrction was stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*day*	Pointer to integer, returns the day of the date
out	*month*	Pointer to integer, returns the month of the date
out	*year*	Pointer to Integer, contains the year of the date
out	*hh*	Pointer to integer, contains the hours of the time after the jump
out	*mm*	Pointer to integer, contains the minutes of the time after the jump
out	*ss*	Pointer to integer, returns the seconds of the time

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.15 GOMDBTS256_substitutionGeoGetDateTime2()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetDateTime2 (
            int handle,
            double * value )
```

This method returns the date and time the current substitution corrction was stored as timestamp.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to double, contains the time as timestamp

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.16 GOMDBTS256_substitutionGeoSetComment()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoSetComment (
            int handle,
            char * comment )
```

This method sets a comment describing the current substitution more closely. It should be called before the current substitution correction is saved, since this comment is also stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*comment*	null-terminated string containing the comment; Maximum permissible length including terminator: 1024 bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.17 GOMDBTS256_substitutionGeoGetComment()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetComment (
            int handle,
            char * comment )
```

This method returns the comment set by the substitutionSetComment method.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*comment*	null-terminated string, must be allocated with 1024 bytes; contains the comment

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.18 GOMDBTS256_substitutionGeoSetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoSetIntegrationTimeInMs (
            int handle,
            int value )
```

With this method it is possible to set the integration time in µs, which is used for the substitution measurement. Value range is 2 - 4000000 -> 2µs bis 4sec. If you have a device with cooling (cooling have to switch ON), it is valid to use a value up to 60000000 µs (60 sec.). If the integration time selected to high, the value is overloaded and not usable.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Integer value, integration time in microseconds [µs].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.19 GOMDBTS256_substitutionGeoGetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetIntegrationTimeInMs (
            int handle,
            int * value )
```

Returns the last selcted integration time in µs.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to integer value, contains the integration time in microseconds [µs].

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.20 GOMDBTS256_substitutionGeoSetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoSetDynamicTimeMode (
            int handle,
            bool value )
```

This method activates the dynamic adjustment of the integration time for the substitution.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Boolean Value: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.12.2.21 GOMDBTS256_substitutionGeoGetDynamicTimeMode()

```
int GOFUNCDESC GOMDBTS256_substitutionGeoGetDynamicTimeMode (
            int handle,
            bool * value )
```

This method returns the dynamic time mode for the substitution.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean; containing the „time mode“: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13 Calibration methods

Functions

- int GOFUNCDESC GOMDBTS256_calibrationLoadFromDevice (int handle)
- int GOFUNCDESC GOMDBTS256_calibrationSaveToDevice (int handle, int calibrationEntryNumber)
- int GOFUNCDESC GOMDBTS256_calibrationIntensitySetCalibLampFileName (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityGetCalibLampFileName (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityMeasureSpectral (int handle, double ∗saturation, double ∗countsDark, double ∗countsSignal, double ∗calFactors)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityMeasureIntegral (int handle, double ∗currentDark, double ∗currentSignal, double ∗calFactor)
- int GOFUNCDESC GOMDBTS256_calibrationIntensitySetIntegrationTimeInMs (int handle, int value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityGetIntegrationTimeInMs (int handle, int ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensitySetCalibrationName (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityGetCalibrationName (int handle, char ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityGetFactorIntegral (int handle, double ∗value)
- int GOFUNCDESC GOMDBTS256_calibrationIntensityGetFactorsSpectral (int handle, double ∗values)

5.13.1 Detailed Description

The following methods are necessary to perform a calibration.

These methods can be called at any time, but the storage method of the calibration configuration is password-protected, as special conditions must be met for calibration. The configuration numbers 10 - 19 are available for saving. The configuration numbers 0 - 9 are reserved for factory calibrations. The different configuration areas are password protected (see method "setPassword"). However, the authorization is only checked when calling the method "calibSaveToDevice".
A detailed description of the procedure can also be found in the documentation: "S-BTS2048 Recalibration Manual.pdf".
Before a calibration can be saved, all necessary data must have been set using the methods listed in this chapter. If only individual data are to be changed in an existing calibration, it is advisable to load the existing calibration, then to reset the data to be changed and then to save the calibration either under a new configuration number or under the old configuration number.
All calibration data is first temporarily stored in a temporary memory and only transferred to the memory of the measuring instrument when the saveToDevice method is called.
The simplest case of a calibration can be done as follows:

1. Load existing calibration (calibLoadFromDevice)
2. Set calibration standard (calibSetCalibLampFileName)
3. Perform self-absorption correction (see Self-absorption correction)
4. Switch on the calibration lamp and wait for the burn-in time
5. Set the integration time for the spectral and integral calibration measurement (calibSetIntegrationTimeInUs)
6. Calibrate spectrally (calibMeasureSpectral)

7. Calibration measurement integrally (calibMeasureIntegral)

8. Define new name for calibration (calibSetCalibrationName)

9. Store calibration in the meter under a configuration number (calibSaveToDevice). An already existing configuration will be overwritten

5.13.2 C++ Calling Example

load old calibration. perform recalibration and save it to device.

```
int handle;
GOMDBTS256_getHandle(NULL, &handle);
       //initialization
char value[32];
GOMDBTS256_calibrationLoadFromDevice(handle, 0);
       //load old calibration data
GOMDBTS256_calibrationIntensitySetCalibLampFileName(handle, "C:\\temp\\lamp.txt
");                     //set path and filename of calib lamp
GOMDBTS256_calibrationIntensitySetIntegrationTimeInMs(handle, 2000);
double saturation, countsDark[256], countsSignal[256], factors[256];
GOMDBTS256_calibrationIntensityMeasureSpectral(handle, &saturation, countsDark, countsSignal, factors);
       //measure spectral
double currentDark, currentSignal, factor;
GOMDBTS256_calibrationIntensityMeasureIntegral(handle, &currentDark, &currentSignal, &factor);
       //measure integral
GOMDBTS256_calibrationIntensitySetCalibrationName(handle, "Test Calib
");
GOMDBTS256_calibrationSaveToDevice(handle, 16);
       //save to config 16
GOMDBTS256_releaseHandle(handle);
       //release handle
```

5.13.3 Function Documentation

5.13.3.1 GOMDBTS256_calibrationLoadFromDevice()

```
int GOFUNCDESC GOMDBTS256_calibrationLoadFromDevice (
              int handle )
```

this method loads the calibration data of the currently selected calibration configuration (“setCalibrationEntry↩ Number”) from the measurement device in the buffer memory. The data can then be read from the buffer memory using the “Get” methods. If one wants to change certain values in a calibration configuration, it is recommended that the existing configuration be loaded first, the desired changes be made and then saved. Loading of calibration data can last up to one minute.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.2 GOMDBTS256_calibrationSaveToDevice()

```
int GOFUNCDESC GOMDBTS256_calibrationSaveToDevice (
            int handle,
            int calibrationEntryNumber )
```

The currently defined calibration data is saved in the measurement device. Before saving, it is checked whether all the required parameters have been set. If not, an error code is returned.
The “user-specific” memory is defined in numbers 15 – 23. The positions 0 – 14 are reserved for factory calibrations by Gigahertz-Optik.
The use of this method is rejected if the given password for calibrations is not accepted. The process of saving can take up to one minute.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value; configuration number under which this calibration is saved in the device, 0 – 14 are reserved for factory calibrations, 15 – 23 are freely available

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.3 GOMDBTS256_calibrationIntensitySetCalibLampFileName()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensitySetCalibLampFileName (
            int handle,
            char * value )
```

Loads calibration data from an external file and uses this data to calibrate. If this method is used, the "calib↩ AzSetCalibLamp" method can not be used. If the calibration factors are to be determined using the calibration measurement methods, this method must be used instead of "calibAzSetCalibLamp". \ n The calibration lamp data must be available in unit [W]. The format for calibration files is as follows:
Line: (optional) Comment line, marked by "//" or ";" at the beginning of the line \ n The following lines: in each line an entry (wavelength and associated lamp value separated by tab) \ N Example: \ n // comment line \ n 250 124.365 \ n 255 166.447 \ n 260 215,786 \ n 265 278,089 \ n ...

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	nullterminated string; contains complete path to the lamp-file

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.4 GOMDBTS256_calibrationIntensityGetCalibLampFileName()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityGetCalibLampFileName (
            int handle,
            char * value )
```

Returns the previously defined calibration-filename with the complete path. If the calibration-lamp data are set using the method "calibAzSetCalibLamp", then the file name is empty.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	null-terminated string; Contains the complete path to the lamp file, the string must be allocated with 2048 bytes.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.5 GOMDBTS256_calibrationIntensityMeasureSpectral()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityMeasureSpectral (
            int handle,
            double * saturation,
            double * countsDark,
            double * countsSignal,
            double * calFactors )
```

With this method the spectral calibration factors can be determined by a calibration measurement. Before this, the calibration lamp must have been set with "calibSetCalibLampFileName".

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*saturation*	pointer to double value; Contains the control of the spectral unit during the calibration measurement
out	*countsDark*	Pointer on the first element of a double array, contains the dark signal of the spectral unit during the calibration measurement; Memory needs to be allocated for 2048 double values
out	*countsSignal*	Pointer on first element of a double array, contains the useful signal of the spectral unit during the calibration measurement; Memory needs to be allocated for 2048 double values
out	*calFactors*	Pointer on first element of a double array, contains the determined calibration factors of the spectral unit at the Calibration; Memory needs to be allocated for 2048 double values

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.6 GOMDBTS256_calibrationIntensityMeasureIntegral()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityMeasureIntegral (
            int handle,
            double * currentDark,
            double * currentSignal,
            double * calFactor )
```

With this method the spectral calibration factor can be determined by a calibration measurement. Before this, the calibration lamp must have been set with "calibSetCalibLampFileName".

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*currentDark*	pointer to double value, contains the dark current of the integral unit during the calibration measurement
out	*currentSignal*	pointer to double value, contains the useful current of the integral unit during the calibration measurement
out	*calFactor*	pointer to double value, contains the determined calibration factor of the integral unit during the calibration measurement

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.7 GOMDBTS256_calibrationIntensitySetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensitySetIntegrationTimeInMs (
            int handle,
            int value )
```

This method defines the integration time to be used in the spectral calibration measurement. If saturation is too low (should be at least 54%), the integration time should be greater.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	double value, contains the integration time in milliseconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.8 GOMDBTS256_calibrationIntensityGetIntegrationTimeInMs()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityGetIntegrationTimeInMs (
            int handle,
            int * value )
```

This method defines the integration time to be used in the spectral calibration measurement. If saturation is too low (should be at least 54%), the integration time should be greater.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	pointer to double value, contains the integration time in milliseconds

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.9 GOMDBTS256_calibrationIntensitySetCalibrationName()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensitySetCalibrationName (
            int handle,
            char * value )
```

This method defines the name of the calibration configuration; this is displayed in the configuration window. The name should be unambiguous so as not to cause confusion. Factory calibrations are always named as follows: “NAME (UNIT)”, whereby NAME can be any random name and UNIT the actual unit of the integral measurement unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
in	*value*	Zero terminated string, max. length: 31 characters plus zero terminator

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.10 GOMDBTS256_calibrationIntensityGetCalibrationName()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityGetCalibrationName (
              int handle,
              char * value )
```

This method returns the previously defined calibration name.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Zero terminated string; contains the configuration name after return, minimum size: 32 bytes

Returns

Integer values; for values unequal to "0", see Errors and Warnings

Example:

```
int handle;
GOMDBTS256_getHandle(NULL, &handle);
char value[32];
GOMDBTS256_calibLoadFromDevice(handle);
GOMDBTS256_calibGetCalibrationName(handle, value);
//do anything with values
GOMDBTS256_releaseHandle(handle);
```

5.13.3.11 GOMDBTS256_calibrationIntensityGetFactorIntegral()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityGetFactorIntegral (
              int handle,
              double * value )
```

This method returns the previously defined calibration factor of the integral unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains the previously defined calibration factor for the integral unit after return.

Returns

Integer values; for values unequal to "0", see Errors and Warnings

5.13.3.12 GOMDBTS256_calibrationIntensityGetFactorsSpectral()

```
int GOFUNCDESC GOMDBTS256_calibrationIntensityGetFactorsSpectral (
            int handle,
            double * values )
```

This method returns the previously defined calibration factors of the spectral unit.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS256; this value is returned by the getHandle method.
out	*values*	values: pointer to the first element of a double array with a size of 256, contains the previously defined calibration factors for all pixels after return -> exactly 256 values

Returns

Integer values; for values unequal to "0", see Errors and Warnings